

**Application of Neural Networks
to Indirect Monitoring of Helicopter Loads
from Flight Variables**

by

Allan B. Cook

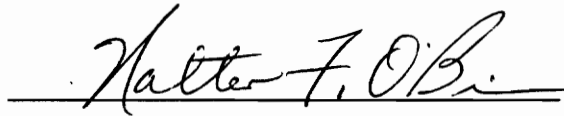
Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER of SCIENCE

in

Mechanical Engineering

APPROVED:



Dr. Walter F. O'Brien, Chairman



Dr. Christopher R. Fuller



Dr. Alfred L. Wicks

October 30, 1991

Blacksburg, Virginia

C.2

LD
5655
V855
1991
C664
C.2

**Application of Neural Networks
to Indirect Monitoring of Helicopter Loads
from Flight Variables**

by

Allan B. Cook

Dr. Walter F. O'Brien, Chairman

Mechanical Engineering

(ABSTRACT)

Many situations arise in engineering where it is desired to model a system of complicated input and output variables. However, analytical difficulties arise when these systems exhibit nonlinear behavior. Neural networks have proven useful for such applications because they are able to model complicated nonlinear systems through exposure to a database including input parameters and the desired outputs. One such complicated system consists of the unknown relationships between flight variables and structural loads on helicopters. The development of an accurate neural network based model would allow indirect monitoring of these loads so that fatigue-damaged components could be replaced according to load history.

In this thesis, an extensive database of real-time flight records has been effectively used to teach a multilayer feedforward artificial neural network nonlinear relationships between common flight variables and the resulting component loads. The trained network predicts time-varying mean and oscillatory load records corresponding to flight variable histories. Component loads in both the fixed and rotating systems of a military helicopter have been resolved over a variety of standard maneuvers. Predictions under the present conditions are on the order of 90 to 100% accurate. Although the range of maneuvers presently considered is rather limited in comparison to the total helicopter flight spectrum, the present results justify further pursuit of this neural network application.

Acknowledgements

First of all, I thank God for providing the opportunity and ability to do this work. Without Him, this would not have been possible.

I wish to thank Dr. Walter O'Brien, my advisor and committee chairman, for his unwavering support and knowledgeable supervision throughout my Masters program. I would also like to thank the other members of my committee, Dr. Chris Fuller and Dr. Al Wicks, for their willing support, advice and expertise.

In addition, I thank Ran Cabell and Emily Scott for being my neural net mentors and sounding boards. Much appreciation is also due to our sponsors at Ft. Eustis Army Base, particularly Randy Buckner, structural aerospace engineer and ex-helicopter pilot who was fun to work with and gave me hands-on helicopter fatigue knowledge.

I especially wish to thank my parents and brother for their encouragement and support throughout my life long. And most importantly, I thank my lovely wife and our children for being my personal fan club and a wonderful distraction throughout this adventure.

Table of Contents

Acknowledgements.....	iii
Table of Contents	iv
List of Figures	vii
List of Tables.....	x
Nomenclature	xi
1 Introduction	1
2 Background	7
2.1 Neural Networks.....	7
2.2 Helicopter Fatigue Problem	11
2.3 Current Helicopter Fatigue Methodology and Maintenance	12
2.4 Efforts to Improve Helicopter Maintenance	14
3 Development of Load-predicting Neural Network.....	16
3.1 Neural Network Approach and Advantages	17
3.2 Helicopter Data	19
3.2.1 Maneuvers Considered	19
3.2.2 Neural Network Input Parameters	23
3.2.3 Fixed System Component.....	24

3.2.4 Rotating System Components.....	27
3.3 Data Management.....	30
3.3.1 Feature selection	31
3.3.2 Data Conditioning.....	32
3.4 Load-predicting Neural Network	37
3.4.1 Neural Network Initialization.....	39
3.4.2 Feedforward Operation.....	40
3.4.3 Back-propagation Training.....	42
3.4.4 Analysis of Network Performance.....	46
4 Demonstration of Load-predicting Neural Network.....	47
4.1 Training and Testing Procedure	47
4.2 Results.....	49
4.2.1 Tailboom Results	55
4.2.2 Pitch Link Results.....	61
4.2.3 Lead-Lag Damper Results.....	66
4.3 Discussion of Results.....	71
4.3.1 Discussion of Tailboom Results	72
4.3.2 Discussion of Pitch Link Results	74
Discussion of Lead-Lag Damper Results.....	75
4.4 Secondary Observations.....	76
4.4.1 Relevance of Input Parameters	76
4.4.2 Variations in Network Geometry.....	78
4.4.3 Variations in Training Specifications	79
5 Conclusions	81
6 Recommendations.....	83

References 85
Vita 87

List of Figures

Figure 1. Generic Neural Network	4
Figure 2. Neural Network Training Analogy to Plant Modeling Scheme	4
Figure 3. Neural Network Application to Indirect Monitoring of Helicopter Loads	5
Figure 4. Feedforward Neural Network.....	8
Figure 5. Sample Processing Unit	9
Figure 6. Typical Fatigue Life Determination Method Leading to Maintenance Schedule Development	14
Figure 7. Helicopter - Plant Analogy.....	18
Figure 8. Sample of Flight Control Step Input.....	20
Figure 9. Step Changes in Stick Position Comprising Seven Maneuvers	21
Figure 10. Transformation from Helicopter to X-Y Coordinates	22
Figure 11. Maneuvers with Adopted Labels	23
Figure 12. Tailboom in Vertical Bending.....	25
Figure 13. Typical Vertical Tailboom Bending Load History.....	25
Figure 14. Sample DFT of Vertical Tailboom Bending	26
Figure 15. Axially Loaded Pitch Link	27
Figure 16. Typical Pitch Link Load History	28

Figure 17. Axially Loaded Lead-Lag Damper.....	29
Figure 18. Typical Lead-Lag Damper Load History.....	30
Figure 19. Load-predicting Neural Network.....	38
Figure 20. Load-predicting Neural Network Operating Schematic	41
Figure 21. Neural Network Training and Testing Procedure.....	48
Figure 22. "Easy"and"Difficult" Maneuvers Selected for Presentation	52
Figure 23. Conditioned Flight Variable Histories for -0.5X "Easy" Maneuver.....	53
Figure 24. Conditioned Flight Variable Histories for +1.0Y "Difficult" Maneuver	53
Figure 25. Tailboom Results. Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "easy")	57
Figure 26. Tailboom Results. Test Maneuver: +1.0Y. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "difficult")	58
Figure 27. Tailboom Results. Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, ____, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-two, "easy")	59
Figure 28. Tailboom Results: Test Maneuver, +1.0Y, Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, _____, -1.0Y (mode-two, "difficult")	60
Figure 29. Pitch Link Results: Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "easy")	62
Figure 30. Pitch Link Results: Test Maneuver: +1.0Y. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "difficult")	63
Figure 31. Pitch Link Results: Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, ____, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-two, "easy")	64
Figure 32. Pitch Link Results: Test Maneuver, +1.0Y, Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, _____, -1.0Y (mode-two, "difficult")	65
Figure 33. Lead-Lag Damper Results: Test Maneuver: -0.5X. Training Maneuvers: +0.5X,	

	+1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "easy")	67
Figure 34. Lead-Lag Damper Results: Test Maneuver: +1.0Y. Training Maneuvers: +0.5X,	+1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "difficult").....	68
Figure 35. Lead-Lag Damper Results: Test Maneuver: -0.5X. Training Maneuvers: +0.5X,	+1.0X, _____, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-two, "easy")	69
Figure 36. Lead-Lag Damper Results: Test Maneuver, +1.0Y, Training Maneuvers: +0.5X,	+1.0X, -0.5X, -1.0X, +0.5Y, _____, -1.0Y (mode-two, "difficult").....	70

List of Tables

Table 1. Original Maneuver Names and Adopted Cartesian Labels.....	22
Table 2. Flight Variables Used as Neural Network Inputs.....	24
Table 3. Input Parameter (Flight Variable) Normalizing Factors.....	33
Table 4. Output Parameter (Component Load) Normalizing Factors.....	34
Table 5. Description of Presented Results.....	54
Table 6. Tailboom Test Results.....	56
Table 7. Pitch Link Test Results.....	61
Table 8. Lead-Lag Damper Test Results.....	66
Table 9. Relevance of Input Parameters.....	77

Nomenclature

Variable Name	Description
d_j	desired output for unit j
\bar{E}	average percent-of-scale error for all cases K
$E(k)$	percent-of-scale error for case k
E_{LMS}	least mean square error
G's	units of acceleration corresponding to 32.2 ft/sec ²
η	gain term, controls severity of weight changes
H_n	output of unit n
H_o	constant input equal to 1
\hat{H}_n	sum of weighted inputs for unit n
LMS	least-mean-square
-1.0X	left lateral cyclic step stick input of 1.0 in.
-1.0Y	aft longitudinal cyclic step stick input of 1.0 in.
-0.5X	left lateral cyclic step stick input of 0.5 in.
<i>offset</i>	offset for scaling variable X
+1.0X	right lateral cyclic step stick input of 1.0 in.
+1.0Y	forward longitudinal cyclic step stick input of 1.0 in.

$+0.5X$	right lateral cyclic step stick input of 0.5 in.
$+0.5Y$	forward longitudinal cyclic step stick input of 0.5 in.
<i>range</i>	difference between upper and lower bounds for scaling variable X
$W1_{mn}$	adaptive weighted connection from unit m to unit n
$W2_n$	adaptive weighted connection from unit n to output unit
X_{MAX}	upper bound for scaling variable X
X_{MEAN}	mean of X for current window
X_{MIN}	lower bound for scaling variable X
x_0	constant input unit, equal to 1
x_m	current value of input m
X_{OSCI}	standard deviation of X for current window
X_{SCALED}	scaled value of variable X
y	current output of neural network
y_j	neural network output of unit j
\hat{y}	sum of weighted inputs to output unit

1 Introduction

Many situations arise in engineering applications where it is desired to model cause-effect relationships between two sets of conditions (inputs and outputs) for a given system. Analytical models allow engineers and scientists to better understand physical systems and processes thus affording virtually unlimited advantages and applications. For example, when the governing equations of a system have been established, one is able to make intelligent decisions about how to control specific variables or how to improve system performance. In other applications, it is desired to predict the results of an event prior to, or without, that event actually occurring. For example, a nuclear power plant could be modeled to predict the outcome of various failures, allowing anticipation of emergencies and the design of proper backup systems. Or, perhaps one wishes to analyze the outcome of a proposed design change. In this case, the model can be modified instead of the physical system, thus avoiding the costs associated with manufacturing and testing.

Another area that requires a form of modeling is that of active and adaptive control. Usually the objective is to actively control some undesired behavior in a system. To accomplish this, the system output needs to be anticipated so that a cancelling phenomena can be properly generated. An example is the active control of sound. Here, a secondary sound source generates acoustic pressure waves of the same frequency as the original source, but 180 degrees out of phase. This creates destructive interference, thus cancelling or reducing the annoying original signal [1]. In still other circumstances, it

is desired to monitor parameters that, for reasons such as cost and environment, are inaccessible. In these situations an accurate model of the relationships between attainable and unattainable variables makes indirect monitoring of the desired variables possible [2]. In summary, an operative analytical model of the cause-effect relationships in any system provides numerous benefits to those people involved.

Today, there is a diversity of techniques available for system modeling. Most can be described as combinations of rule-based, empirical, statistical, and theoretical methods. Perhaps the most elementary models are "rule-based". These models operate like flowcharts and are based on observations or desired performance. Recently they have evolved into what are called "expert systems". These systems seek to immediately benefit the user with a wealth of knowledge gathered from a conglomeration of sources. Two other closely related areas are statistical and empirical modeling. These type of models use correlation schemes and existing data or experience to resolve effective relationships between parameters. Simple examples include least-squares-fit and linear regression. The most popular type of models, and usually preferred, are theory-based models. These classical models are grounded in theoretical understanding of the underlying mathematics and attempt to account for all the relevant details. It is well worth mentioning that this list is not exhaustive but that most models are combinations of these and other techniques.

However diverse the realms of modeling are, there are still many systems which have proven difficult to model. These limitations are manifest when the response of the physical system and the modeled response diverge or disagree, given the same input. There are many reasons why systems are difficult to model accurately but most are due basically to an incomplete understanding or inclusion of the underlying relationships. Generally speaking, many systems are effected by an overwhelming number of complex interrelated variables. Complexities increase when these systems exhibit nonlinear behavior; many systems only behave linearly and according to theory over a small range of conditions. Thus, accurate relationships are often difficult to determine, and the need for methods of modeling complex multivariable nonlinear systems is established.

A quickly emerging technology that has shown potential for emulating such systems is called adaptive and intelligent modeling. These models attempt to "learn" realistic relationships from experience or exposure to examples, the same way humans often learn. This is in contrast to traditional methods which are usually pre-programmed and theory-based. More specifically, neural networks is a type of adaptive and intelligent modeling that is based on the biological understanding of the brain in hope of simulating its capacity to learn, or model systems. In recent applications, these networks have successfully modeled fairly complicated nonlinear systems without any foreknowledge of the physics involved [3]. This is analogous to a child learning to ride a bicycle without an in-depth understanding of the physics involved. The child learns by trying different inputs and taking note of the responses over and over until an effective model or understanding of the input/response relationships has been encoded in the mind. Neural networks attempt to operate in the same manner; learning from experience without having any preconceived formulation of the problem being solved.

Another way of seeing neural networks is as a multidimensional nonlinear regression scheme that learns the proper weighting functions through intensive repetition of input patterns or vectors and the corresponding correct output patterns or responses. Therefore, a database having both inputs and corresponding outputs is required for "training".

Like the brain, neural networks are composed of a large number of very simple processing units densely interconnected and operating in parallel, as shown in Figure 1. Processing units sum their inputs and produce a corresponding nonlinear output, similar to neurodes in the brain. Connections, like synapses, weight their inputs and feed the information forward to units (or neurodes) in the following layer. Before training, the network has a random nonlinear mapping of input to output variables.

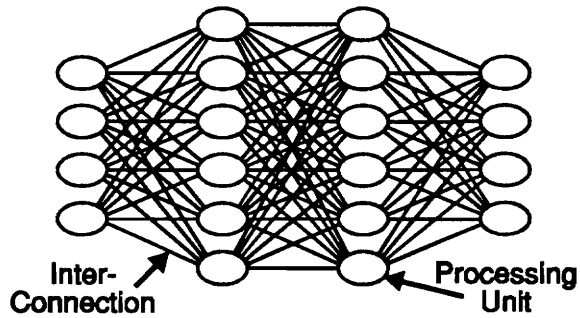


Figure 1. Generic Neural Network

Much like a feedforward controller or the typical plant modeling scheme shown in Figure 2, neural networks are "trained" by comparing network output to plant output for the same input. The error, or some function of the difference, is then fed back to the model, here a neural network, for adaptation of the internal mappings to generate a more accurate response. Adaptation is usually accomplished with an algorithm that seeks to minimize the least-mean-square error. When the network and system responses agree over the range of desired conditions, the network has "converged" to a solution or encoded an effective nonlinear model of the system. Thus, given a suitable database for training, neural networks offer the potential to devise effective nonlinear models where traditional techniques are not satisfactory.

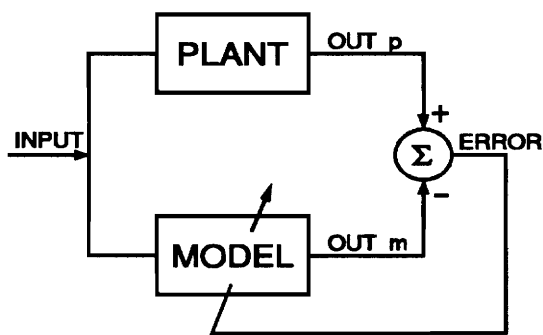


Figure 2. Neural Network Training Analogy to Plant Modeling Scheme

One complex nonlinear system that has proven very difficult to model with traditional methods is comprised of the relationships between standard helicopter flight variables and the loads experienced by critical components, particularly in the main and tail-rotor blades systems. Much effort has been made towards deciphering these relationships but success has been limited by factors including the large number of interrelated variables involved, the inherent complexities of rotary-wing flight, and the sheer volume of data required to span the helicopter flight envelope. Due to the complications of heavily instrumenting aircraft with strain-gauges and transmitting data from a rotating to a fixed environment, direct monitoring and recording of loads is not feasible. The modeling alternative is to infer critical component loads, particularly those in the rotating system, with a nonlinear model that monitors easily measured flight variables in the fixed system, as shown in Figure 3. This hypothetical figure shows load histories inferred from flight variable histories; the ultimate objective of this work.

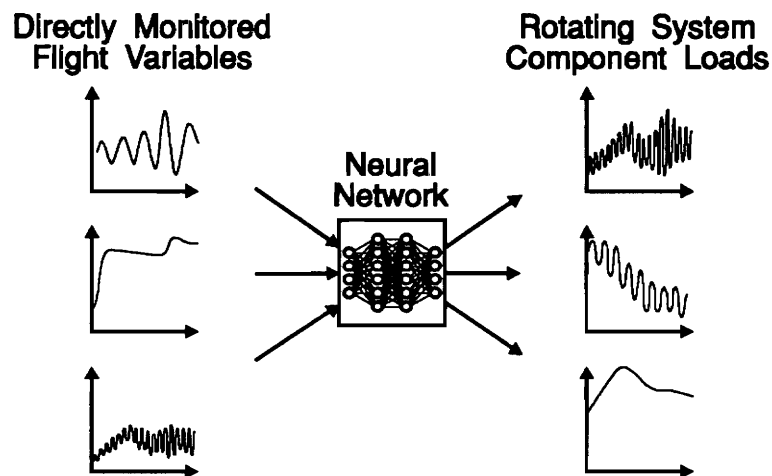


Figure 3. Neural Network Application to Indirect Monitoring of Helicopter Loads

This method of indirectly monitoring loads would facilitate inspection and/or replacement of critical components according to load histories, rather than fixed intervals. Fatigue damage assessment could then be carried out with existing methods, thereby improving reliability and reducing costs [2].

As mentioned previously, suitable neural network models are developed with extensive, representative databases. In this thesis, data collected from a military attack helicopter including flight variable histories and the simultaneously measured loads has been used to develop a neural network model of the complex nonlinear relationships. The load signals have been resolved into components of time-varying mean and oscillatory components for simplicity. The present neural network, having three layers, has successfully inferred mean and oscillatory load histories from flight variable records for three dynamically loaded components over a range of standard maneuvers. This first-time application demonstrates the neural network's ability to model highly complex nonlinear systems through exposure to a suitable database. More particularly, the end result of the work presented in this thesis is the qualified recommendation to continue pursuing a neural network solution for the helicopter fatigue problem.

2 Background

This chapter serves first as an introduction and brief overview of artificial neural networks; in particular, multilayer feedforward neural networks trained with the popular back-propagation technique. Latter sections more fully describe the helicopter fatigue problem and current helicopter fatigue methodology along with a brief account of various efforts to improve the situation.

2.1 Neural Networks

Neural networks represent a technique for developing relationships between input and output parameters where the true relationships are unknown or too complicated to model analytically. Like much of today's technology, neural networks are modeled after a biological phenomena; in this case, the human brain. The physical similarity is the dense interconnection of a vast number of simple processing units operating in parallel as shown in Figure 4 [3]. These units, like neurons in the brain, are stimulated by their inputs to provide a nonlinear output. The interconnections (called synapses in the brain), modify and transmit output from one unit to another based on the connection strength. The resulting network functions similarly to the brain by recognizing an input pattern or stimulation that leads to a corresponding output pattern or conclusion. Another brain-like characteristic of neural networks is the ability to make conclusions when some information is not available or the information is fuzzy [4]. In this sense, neural networks are able to generalize or give a partial solution when other analytical or rule-based systems would fail.

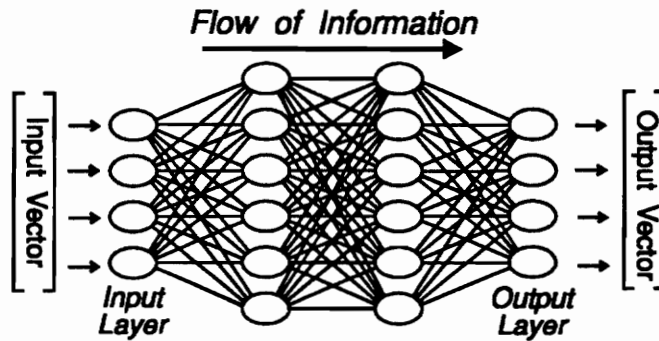


Figure 4. Feedforward Neural Network

Several neural network algorithms are available but most have several attributes in common: they consist of several simple nonlinear processing units interconnected in parallel, reminiscent of the brain; they require some type of training process, whether it be supervised or unsupervised; and they are useful for pattern recognition and/or classification type problems. Selecting a neural network for a particular application includes choice of network type and geometry, choice of training algorithm, and selection of input and output parameters [5]. Training algorithms which require a "teacher" or some type of reference data for training are commonly referred to as supervised learning algorithms [4]. The rest of this thesis applies only to the supervised learning algorithm termed back-propagation.

In normal operation, the input layer units take on values from an input vector (these units do not modify the input vector). The connections weight the inputs and transmit them to the processing units in the next layer. Processing units sum inputs plus a threshold value and produce a nonlinear output based on this sum. The output of these units then fans out through weighted connections to processing units in the following layer. As shown in Figure 4, the information from the input vector "flows" forward from the input layer to the output layer in a forward feeding fashion, Hence, the name "feedforward" neural network. The resulting stimulation levels of the output layer unit are the output vector.

The nonlinear function of the processing units mimics the "firing" action of neurodes in the brain. When neurodes receive enough stimulus, they fire a small electric impulse. Figure 5 shows a

typical processing unit with a summer and sigmoid output function. It is seen that the output is bounded such that a large input or stimulus causes a maximum output (firing) while a small input produces a null output. Many variations in the mathematical formulation of this function exist but most bound the output between zero and one or minus one and one.

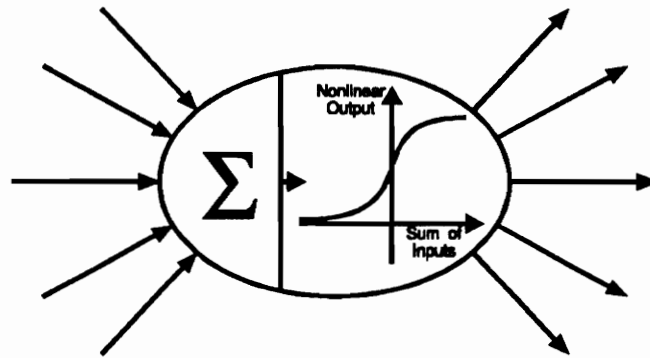


Figure 5. Sample Processing Unit

During training, the network is repeatedly presented with input and the corresponding output data. The internal weights and thresholds are adapted after each trial so as to minimize the squared difference between the network output and the correct or desired output. The method of adjusting these weights depends on the training algorithm. Typically the error is divided among the connections from the preceding layer and individual weights in that layer are adjusted based on their strength or contribution to the error. The severity of adjustment for individual connections is proportional to the perceived contribution to the error. This process continues with units and connections in preceding layers until the entire network has been updated for this trial; hence the name "back-propagation". In this way, the network will more closely provide the correct output given the same input vector. The training process is complete when the network gives satisfactory responses to all training data or meets some minimum acceptable error criteria. At this point, the network is said to have converged to a suitable model and the geometry and weights are stored. It is worthy to note that the network will have only learned the characteristics spanned or contained within the training data. This is, again, due to the fact

that the neural network method is not model-based. Experience has shown, however, that neural networks are fairly adept at interpolating or generalizing within the range of training data and in some cases are able to extrapolate results [6].

The next logical step is quantifying the network performance with test cases. In testing, the network is "shown" input data and produces the modeled response which is then compared to the known or desired system response. Again, the proper output must be known beforehand. Testing data can include both training data and data which has never been presented before. If the user is convinced that the network has satisfactorily modeled the input/output system, then the network is ready to be used in cases where the true output is unknown.

Most neural network applications are classification type problems. In these problems, a set of input features or descriptors leads to a corresponding conclusion. For example, based on such features as hair color, size, complexion, and tone of voice, people are able to conclude the identify of another person. Similarly, neural networks are able to arrive at conclusions from input features based on experience. Successful applications include adaptive noise canceling, mortgage risk evaluation, and bomb sniffing [7].

In addition to classification problems, neural networks can be applied to continuous-value problems. By taking a slice in time, the values of each input variable form a pattern that leads to the corresponding output pattern. The trained network is able to predict an output wave-form according to the time-history of the input variables, similar to the system shown previously in Figure 3. It is noted, however, that this requires significant pre- and postprocessing of data to obtain the desired system performance.

The neural network concept has been around for several decades but growth was severely restricted by lack of interest during the '60's and '70's. This was due mainly to the absence of effective training algorithms, but growth was also slowed by the focus on biological modeling of the brain rather than application to practical problems. Recently though, with the evolution of powerful PCs and the

development of the effective back-propagation training technique and others, there has been a shift in emphasis to practicality. The term neural networks has now become a buzz-phrase with which nearly all computer enthusiasts are somewhat familiar. Within just the past 8 years or so the PC market has been flooded with neural network articles, software packages, and hardware [7].

2.2 Helicopter Fatigue Problem

Helicopters are characterized by extensive vibrations and fatigue damaging structural loads. This is chiefly due to the main-rotor blades which compose a very large, nonuniform inertial system that is constantly changing shape and orientation during flight. Centrifugal forces at the root of the test helicopter's blades are reported to be on the order of 65,000 lbf per blade [8]. Because the blades are discrete and independent of each other, forced changes in orientation of the rotary system result in the generation of extremely large loads in both the fixed and rotating systems.

Fatigue life estimations for dynamic helicopter components are often inaccurate due to the wide variety of flight conditions an individual aircraft may experience. The shortcoming of most fatigue life estimating methods is the fact that they do not account for the flight history of individual aircraft: An aircraft which engages in combat is likely to experience maneuvers of greater intensity than anticipated [9]. The induced structural loadings from such maneuvers can severely reduce the operative life of overloaded components. Cyclic exposure to this type of loading weakens the component materials and is termed fatigue. Due to the light weight requirements of competitive helicopter performance, individual components are designed with relatively short fatigue-life expectancies. This light weight requirement combined with the extreme variations in loading histories results in costly maintenance schedules including frequent inspections and premature retirement of functioning components [10].

In this thesis, critical component refers to a dynamically loaded component typically in the main-rotor or tail-rotor blades system that is necessary for sustained flight. These components, having finite fatigue lives, are identified as critical because most major incidents of rotary-wing flight are related to their fatigue failure. More specifically, several dynamic components on the US Army's Apache AH-64

Attack Helicopter have been labeled critical because their failure implies catastrophic failure for the entire aircraft [10].

Flight records of cyclic loads on 24 of these components and simultaneous measurements of 32 flight variables have been provided through the Ft. Eustis Army Directorate, Ft. Eustis, Va. Included are standard maneuvers as well as simulated air-to-air combat records of the AH-64 engaging other military Attack Helicopters [9]. This extensive database is ideal for examining the potential of applying neural networks to the helicopter fatigue problem. A more thorough description of the data is given in reference 9.

2.3 Current Helicopter Fatigue Methodology and Maintenance

The ideal helicopter maintenance program would result in on-condition retirement of dynamically loaded components just before they fail [11]. In reality, though, current maintenance programs are characterized by frequent inspections and premature replacement of operative components to prevent failures [12]. These programs depend largely on the number of flight hours logged, rather than actual component load histories. Therefore, inspection and replacement of components is inconsistent with actual fatigue damage. In addition, these programs rely on estimated fatigue safe lives for components that are often composed of newly developed composite materials and configurations, the behavior of which relatively little is known [10].

The results of a hypothetical fatigue life problem serve well as a testimony to the difficulty in estimating component fatigue life:

"The hypothetical pitch link problem which was posed to the helicopter industry through the auspices of the American Helicopter Society in October 1979 produced a broad range of results (...). The example problem, which had been very carefully formulated and presented to the industry, was a comparatively simple example of the complexity which often faces the fatigue analyst. Clear definitions of the fatigue test results, flight load measurements and flight spectrum were provided in a concise

format. In spite of this, some interpretations and choices were still available to the analyst in conducting the calculation. The resulting lives for the basic calculation ranged from 9 hours to 2,594 hours (...). The publication of these results simply confirmed the existence of the dilemma that has faced certifying agencies relative to component fatigue life determination." [11]

Thus one can conclude that unpredictable load histories and difficulties in estimating fatigue lives are cause for the instillation of the rigorous, expensive maintenance programs currently in use.

As shown in Figure 6, typical helicopter component fatigue life determination methods generally consist of four steps: (1) Mission Spectrum Definition, (2) Laboratory Fatigue Strength Characterization, (3) Flight Strain Survey, and (4) Safe-Life Calculation. Mission spectrum definition is the characterization of the flight patterns a specific group or fleet of helicopters will experience in normal usage. Defining the mission spectrum classifies the types of maneuvers and their expected frequency during operation. Laboratory fatigue strength characterization is the controlled loading of newly designed components to develop accurate S-N curves. Test helicopters are then fully instrumented with strain gauges and flown through extensive series of maneuvers while recording the resulting component loads. These records are later downloaded into fatigue-damage calculating computer algorithms which utilize the newly developed S-N curves. After determining the respective damage per maneuver, safe-life estimates are made for each component according to the mission spectrum definition. The degree to which these estimates are conservative depends on several factors including acuteness of the component, the variation in expected flight missions, and the economics of regular inspections and replacements. After making safe-life estimates, maintenance schedules are developed for fleets or groups of helicopters having the same mission spectrums so as to minimize failures and sustain flight-ready status [12].

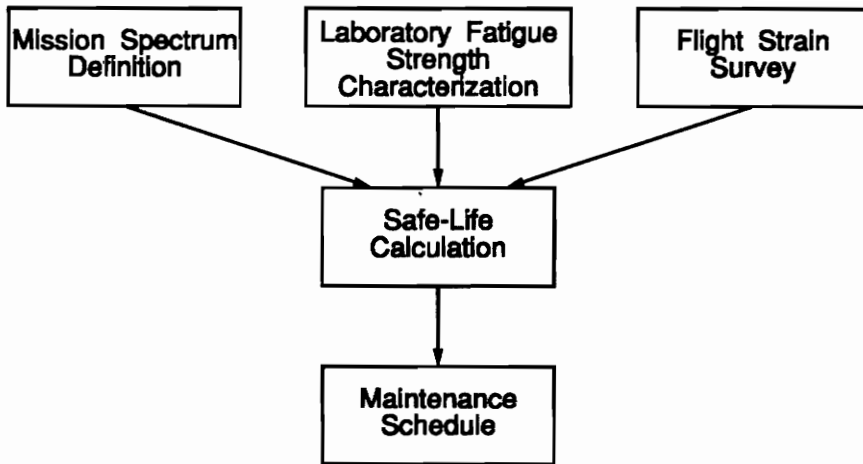


Figure 6. Typical Fatigue Life Determination Method Leading to Maintenance Schedule Development

2.4 Efforts to Improve Helicopter Maintenance

There has recently been a strong interest in reducing helicopter maintenance costs and increasing reliability and fleet readiness in both the military and private sectors. One major division of effort has been the development of fatigue damage monitoring systems. These systems range in complexity from monitoring only mission types to much more expensive systems which monitor loads directly. Major difficulties incurred with the mission type method are classifying mission types and characterizing individual component fatigue damage per mission. The brute force method of monitoring loads directly is usually not feasible due to the costs associated with heavily instrumenting individual helicopters and transmitting the signals from the rotating blades system to an on-board flight data recorder [2].

A more sophisticated approach that lies between these extremes has recently received much attention due to the rapid increase in available computational power. The approach is to correlate commonly monitored flight variables with the more difficult to measure safety-critical loads. Many flight variables are already monitored with current technology at reasonable costs. The challenge is to accurately relate the history these variables to fatigue damage on critical components. More specifically,

the objective is to relate commonly monitored fixed-system variables to loads in the rotating blades system. At first this appears to be an analytical problem of structural dynamics and aerodynamics, but the relationships are extremely complicated and elusive. The development of effective relationships would allow indirect monitoring of rotating system loads through fixed system flight variables.

Similar systems have been implemented successfully in fixed-wing military aircraft using "G" meters and control position indicators to deduce airframe loads throughout the structure and consequently record load histories and peak loadings. This has allowed on-condition retirement of critical components as well as knowledgeable spare parts planning, thus reducing costs [11].

On the other hand, the variables contributing to the dynamic loads on helicopters are greater in number, thought to be nonlinear, and highly interrelated. The predominant loading modes are very complex, consisting of rotor, engine, and transmission excitations from the rotating system interacting with the structural mobility characteristics of the helicopter airframe and control system. For these reasons it has not been possible to transform the output of a simple fixed system array of outputs to load histories of rotating system components [11]. Other popular approaches are sophisticated signal processing and statistical correlation techniques. Typical systems are variations of linear regression models but their success has been limited. Among reasons are the many degrees of freedom in the helicopter flight envelope and the large number of interrelated variables that contribute to the resolution of a given load.

3 Development of Load-predicting Neural Network

There are many steps involved in the development of a neural network for a specific application. First, the need for a neural network solution must be established. If other methods exist and are satisfactory, then this new approach is hard to justify. The second step is choosing a network type and training algorithm. Another requirement is a suitable database. In many situations, data have been gathered for years without gaining a satisfactory understanding of the underlying relationships. This is the ideal situation for a neural network application. If, however, a traditional model exists which partially models the system, then a possible alternative is incremental modeling. Here, the neural network model works in parallel to the existing model and only improves the overall performance as it captures the nonlinear behavior. Thus, the ideal situation is one where data is available or can be easily obtained, existing techniques have proven unsatisfactory, and the relationships are thought to be complex and nonlinear [6].

After establishing the usefulness of a neural network solution, the next step is approaching the problem. This involves determining how to use a neural network in the modeling scheme, which may require creativity and innovation. Finally, one is ready to begin acquiring and preparing data, and then developing and tailoring an appropriate network.

This chapter begins by introducing the neural network approach to indirect load monitoring and the many inherent advantages of this method. The second section briefly describes the three critical

components whose loads have been predicted and the eight flight variables which were used to infer the loads. Also described are the maneuvers from which data was taken to train and test the network. Section three discusses "feature selection" (the choosing of relevant input parameters) and reports how the data was conditioned for use with a neural network. The last section describes the load-predicting neural network in detail including initialization, feedforward operation, training, and analysis of performance.

3.1 Neural Network Approach and Advantages

The neural network approach to load predicting is to model the nonlinear system of standard flight parameters and the resulting dynamic loads with a multilayer feedforward neural network. This approach is analogous to the familiar plant modeling scheme shown in Figure 7 where the goal is to model the response of a system or plant over a variety of input conditions. Typically, in modeling a plant, a theory-based model is invoked and then iteratively modified by comparing the plant and model outputs for the same inputs until satisfactory agreement occurs over the desired range of data. Figure 7 shows the helicopter as analogous to a plant, producing loads from flight variable inputs, and a neural network as the helicopter model. The network learns effective relationships through an iterative training process using loads data from instrumented aircraft engaging in standard maneuvers. The resulting network resolves nonlinear dynamic loads by monitoring available flight variables.

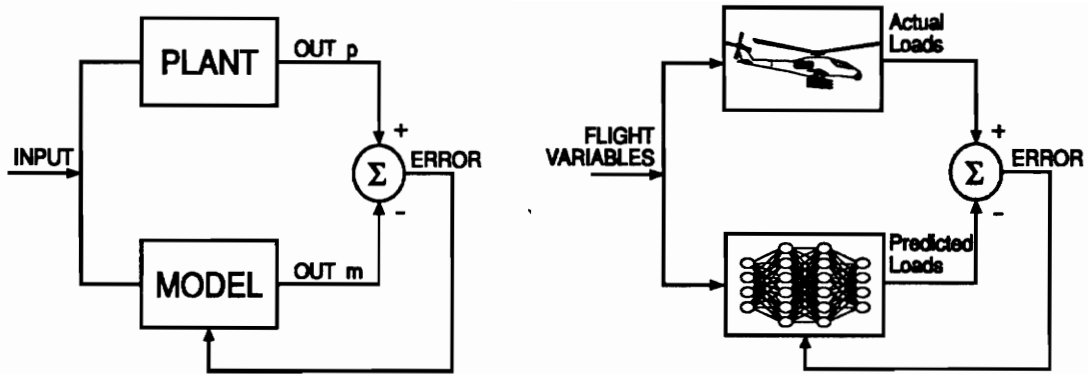


Figure 7. Helicopter - Plant Analogy

After successfully modeling the flight variables/loads system, the trained network could be hard-wired into a simple parallel processing chip which would receive real-time flight data and determine the corresponding loads. The output could then be recorded and postprocessed to determine accumulated fatigue damage per component. This system could be incorporated into an on-board "fatigue monitor" which maintenance personnel would plug into after flight to check the status of safety-critical components. Another option would be to record flight variables with existing flight recorders and download flight data to a stationary neural network algorithm that reproduces load histories and keeps track of fatigue damage for individual helicopters in a fleet.

There are several reasons why the neural network approach is favorable for predicting dynamic helicopter loads. First of all, neural networks learn input-output relationships through experience rather than using predetermined theoretical equations that are not fully developed. It is also worthy to note that training can be an ongoing process such that the network adapts to changes in the system. Neural networks can also select relevant parameters from a large group of inputs. Another characteristic of neural networks which surpasses traditional modeling schemes is the ability to effectively model nonlinear systems. This trait is inherent because the output of individual processing units is typically a nonlinear function of the weighted sum of the inputs. Finally, neural networks are suitable for this

application because of their ability to generalize and make decisions or predictions when information is fuzzy or incomplete. In summary, neural networks have several properties which offer the potential to devise an effective nonlinear model to best predict the critical component loads.

3.2 Helicopter Data

The data used to develop the present network is a small subset of flight recordings taken from the US Army's AH-64 helicopter undergoing standard maneuvers. This extensive database consists of 24 flight variables and 32 component loads for a total of 56 parameters monitored in time with a maximum sample rate of 470 Hz. A thorough description is available in reference 9. Although this database is extensive, having several flight records as long as 60 seconds, the present network was developed using only 21 seconds of data which come from seven standard maneuvers. In addition, only eleven of the 56 available parameters have been used (eight flight variables and three loads). Admittedly, this subset of data represents only a small sample of the entire flight regime but it is hoped that the principles developed here will extend as the envelope of data expands and the methods are refined. This section describes the envelope of seven maneuvers considered, the eight flight variables used, and the three critical components loads which have been successfully modeled.

3.2.1 Maneuvers Considered

The basic controls for maneuvering a helicopter are: 1) collective pitch, which controls the amount of lift from the main-rotor blades, 2) stick position, which controls the aircraft pitch and roll, and 3) pedal positions, which manipulate tail-rotor thrust to control aircraft yaw. The subset of data used in developing the network presented in this thesis consists of seven standard maneuvers which are step changes in stick position while in steady state forward flight. This means the pilot simply moved the control stick quickly in one direction a specific amount from the steady-state forward flight position. All other flight controls remained approximately fixed. Figure 8 shows the lateral stick position history for one of the seven maneuvers; a one-inch left stick input. This flight record, like the others is three

seconds long. Here, it is seen that one-inch corresponds to about 15% of the entire range of motion in the lateral direction from full left to full right. It is also noted that, after the step change, the pilot gradually returned the stick to the neutral position. This is fairly typical of the other maneuvers, which are either half- or one-inch step changes in stick position in varying directions.

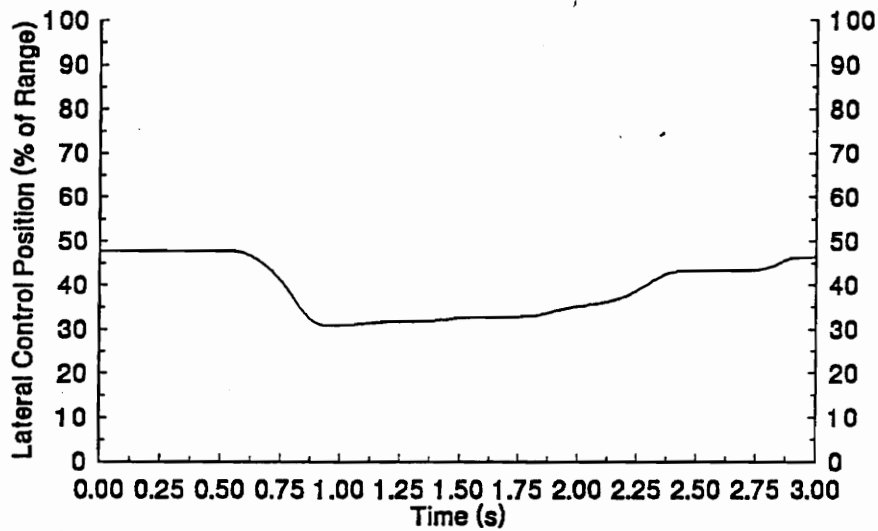


Figure 8. Sample of Flight Control Step Input

Figure 9 shows the changes in stick position in the form of a diagram. Again, each maneuver is a step change in stick position from the steady-state forward flight position. As shown in the diagram, the maneuvers differ from each other in either direction, magnitude, or both.

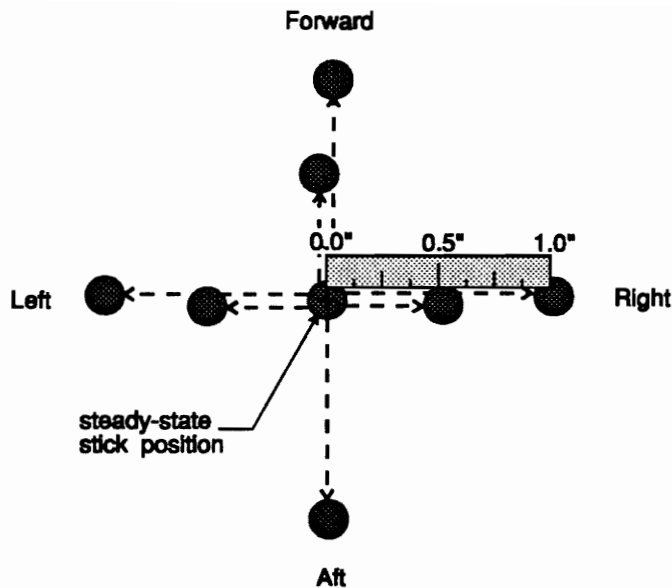


Figure 9. Step Changes in Stick Position Comprising Seven Maneuvers

In helicopter terminology, these maneuvers are referred to as lateral or longitudinal cyclic step maneuvers. The reason being that a change in stick position causes the individual main-rotor blades to change their pitch cyclically as they rotate about the hub. For example, a left lateral cyclic step causes the pitch of each blade to minimize as it passes over the left side of the aircraft and to maximize over the right side, causing the helicopter to roll to the left. Thus, these maneuvers induce large cyclic loads on components throughout the structure [10].

To aid in referencing the maneuvers in the rest of the document, they have been labeled in an X-Y coordinate manner. Figure 10 shows the transformation from helicopter stick position coordinates to the new Cartesian coordinates. Table 1 lists the original maneuver names with the adopted labels. Finally, Figure 11 shows the maneuver diagram with the new labels as they will be referred to in the rest of this document.

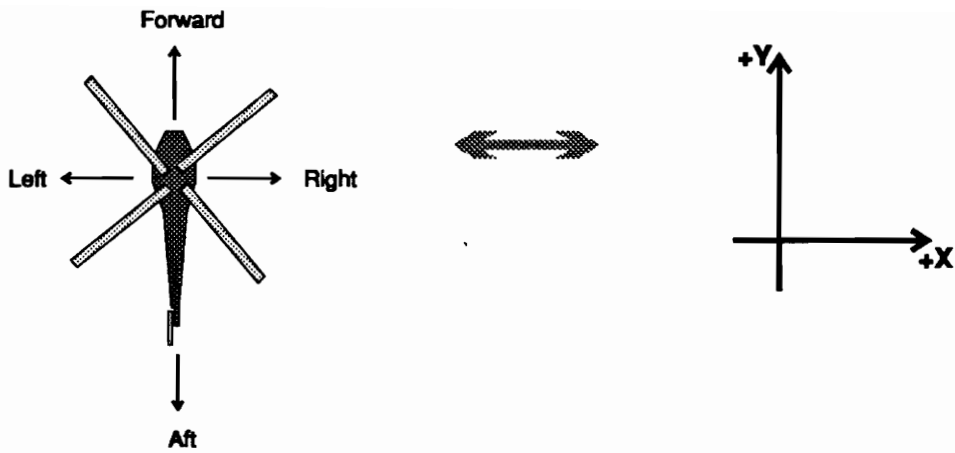


Figure 10. Transformation from Helicopter to X-Y Coordinates

Table 1. Original Maneuver Names and Adopted Cartesian Labels

Maneuver	Label
Right Lateral Cyclic Step, 1/2 in.	+0.5 X
Right Lateral Cyclic Step, 1 in.	+1.0 X
Left Lateral Cyclic Step, 1/2 in.	-0.5 X
Left Lateral Cyclic Step, 1 in.	-1.0 X
Forward Longitudinal Cyclic Step, 1/2 in.	+0.5 Y
Forward Longitudinal Cyclic Step, 1 in.	+1.0 Y
Aft Longitudinal Cyclic Step, 1 in.	-1.0 Y

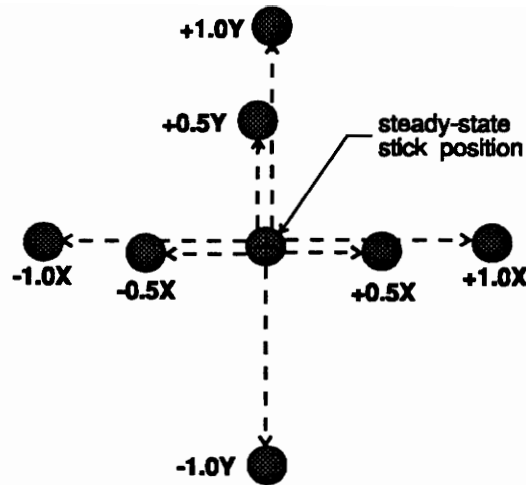


Figure 11. Maneuvers with Adopted Labels

3.2.2 Neural Network Input Parameters

A neural network must have sufficient input information to effectively predict the desired output information. For the helicopter loads application, input information consists of those parameters which are fairly easy to monitor. Thus, neural network input parameters are limited to the commonly monitored flight variables. These include variables such as the pilot's control positions (or inputs to the helicopter) as well as resultant variables like pitch, roll, and yaw rates. The desired output variables are the loads on critical components, which are not feasible to monitor directly.

In the current application, eight flight variables have been selected as neural network inputs. These selections were based on knowledge of the loads system, engineering judgment, and availability [13]. In other words, it was believed that these flight variables would contain the information necessary to infer the critical component loads of interest. Table 2 lists the network inputs. The parameter names are self-explanatory.

Table 2. Flight Variables Used as Neural Network Inputs

Input #	Parameter	Units
1	Pitch Rate	deg/sec
2	Roll Rate	deg/sec
3	Yaw Rate	deg/sec
4	Vertical G's	G's
5	Lateral G's	G's
6	Longitudinal G's	G's
7	Longitudinal Control Position	Percent
8	Lateral Control Position	Percent

The selection of input parameters is often termed "feature selection" in neural network terminology. This refers to the selection of distinguishing input features which lead to some type of conclusion about the output variable, such as identity or magnitude. A latter section in this chapter, 3.3.1, further discusses feature selection and its pertinence to the helicopter loads application.

3.2.3 Fixed System Component

The current neural network has been used to predict loads on three critical components; one in the fixed system and two in the rotating system. Here, fixed system refers to structural components which are stationary relative to the airframe. Rotating system components are those which rotate with the main-rotor blades, including driving and driven components. The first component load modeled was bending in the vertical plane of the tailboom. This particular load was selected as a starting point because it is part of the fixed system, and it's loads would therefore be easier to relate to flight variables [13]. The tailboom supports the tail-rotor which offsets the torque of the main-rotor and thus balances the aircraft as a whole. The critical loading mode considered for the tailboom structure is bending in the

vertical plane, which, for the maneuvers considered, ranged from about -190,000 to -70,000 in-lbf.

Figure 12 shows the approximate location and sign convention used in measuring this load [9].

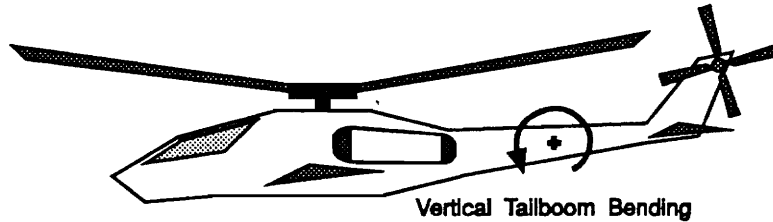


Figure 12. Tailboom in Vertical Bending

Fatigue damage to the tailboom is caused by a combination of transient and cyclic loads. Figure 13 shows a typical trace of the vertical tailboom bending load. A representative data sample for the tailboom was high frequency filtered to determine and remove the time-varying mean (or transient) constituent of the load. It was discovered that the range of time-varying mean loads is larger than the cyclic load amplitude range.

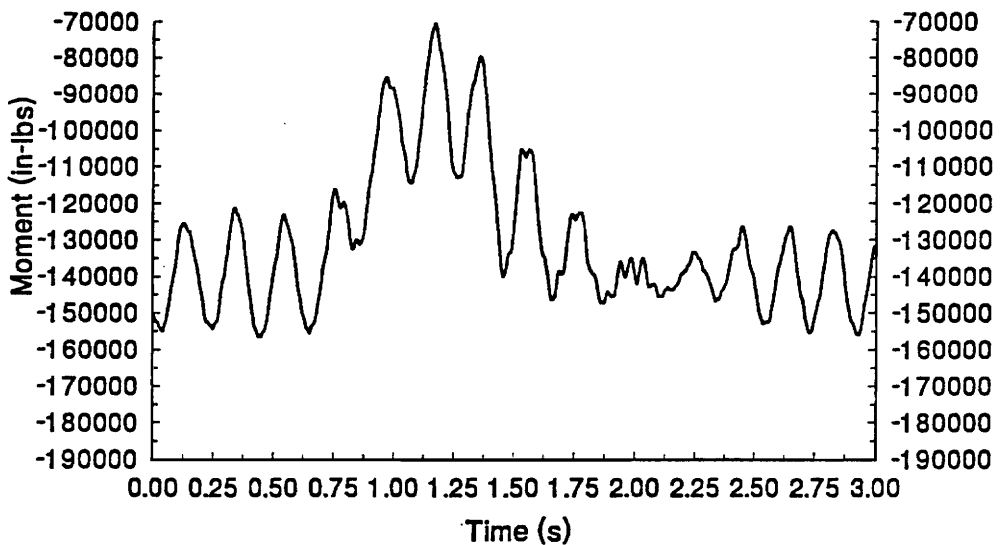


Figure 13. Typical Vertical Tailboom Bending Load History

The transient nature of the mean vertical tailboom bending load is attributed to vertical accelerations and changes in the pitch rate of the aircraft which induce noncyclic bending loads in the tailboom. Also, in forward flight, changes in orientation of the aircraft cause transient aerodynamic loads on the fuselage. For example, when accelerating in the forward direction, the helicopter pitches forward which results in an increased downward aerodynamic load on the tailboom [10].

A sample DFT of the mean-zeroed data, as seen in Figure 14, shows two principal cyclic loading constituents at 4.75 Hz and 19 Hz. The 4.75 Hz constituent is the larger of the two and corresponds to the helicopter main-rotor frequency. This cyclic load is attributed to gyroscopic effects induced when the plane of the main-rotor is changing. The smaller constituent of 19 Hz is a harmonic of four times the main-rotor frequency and, due to the four-blade configuration of the test helicopter, corresponds to the blade passing frequency. This higher frequency constituent is believed to reflect both the transmission of individual blade loads to the fixed system as their pitch and orientation change cyclically, and to pressure waves from the blades passing over the tailboom [10].

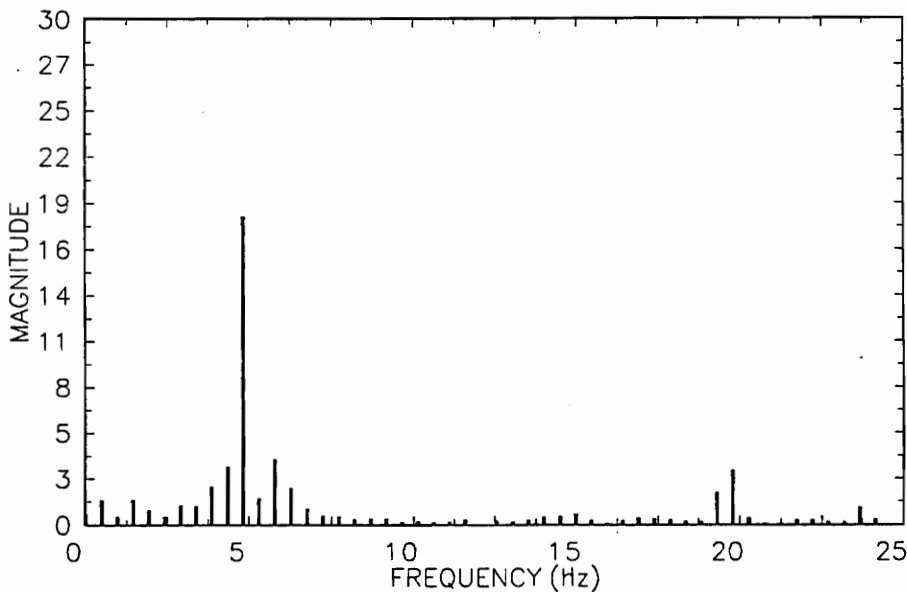


Figure 14. Sample DFT of Vertical Tailboom Bending

3.2.4 Rotating System Components

The other two components considered, a pitch link and a lead-lag damper, are located in the rotating system where damaging loads are chiefly cyclic [10]. Figure 15 is a simplified representation of a pitch link. As seen in the figure, pitch links are loaded axially as they control the pitch of individual main-rotor blades [10],[14]. For the seven current maneuvers, the pitch link load ranged from approximately -400 to 400 lbf.

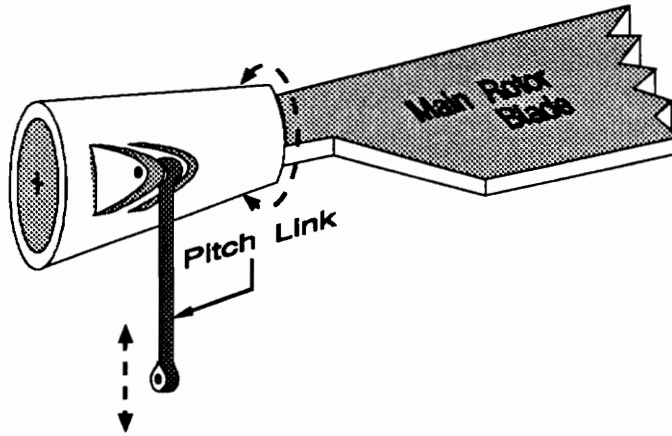


Figure 15. Axially Loaded Pitch Link

A collective pitch change causes uniform increase or decrease of pitch in all four blades simultaneously and results in a proportional change in the time-varying mean load on the pitch links. Collective pitch changes effect the overall lift of the main-rotor blades system. As mentioned earlier, the present data set contains no collective pitch changes but rather cyclic pitch changes due to lateral and longitudinal control stick inputs.

On the other hand, cyclic changes in pitch effect the pitch of blades individually as they rotate about the main-rotor hub. This type of maneuver results in increased cyclic loading of the pitch link. For example, a forward change in cyclic pitch causes the pitch of each blade to minimize over the front of the helicopter and maximize as it passes over the tailboom, causing the aircraft to pitch forward. Another example of cyclic pitch change occurs in steady-state forward flight. Under these conditions, each blade

experiences a change in relative air speed as it cycles through advancing and retreating states. The pitch must therefore also change cyclically to balance the lifting forces.

Similar to the tailboom bending load, the pitch link loads are composed of two principal sinusoidal constituents located at one and four times the main-rotor frequency. The one-per-rev constituent corresponds to the cyclic changing of pitch while the four-per-rev represents interaction with the other three blades [10]. Figure 16, a typical pitch link load history from the maneuvers considered, shows there to be little change in the time-varying mean pitch link load. This reaffirms that each of the seven maneuvers considered is a step change in the cyclic pitch of the main-rotor blades rather than collective pitch.

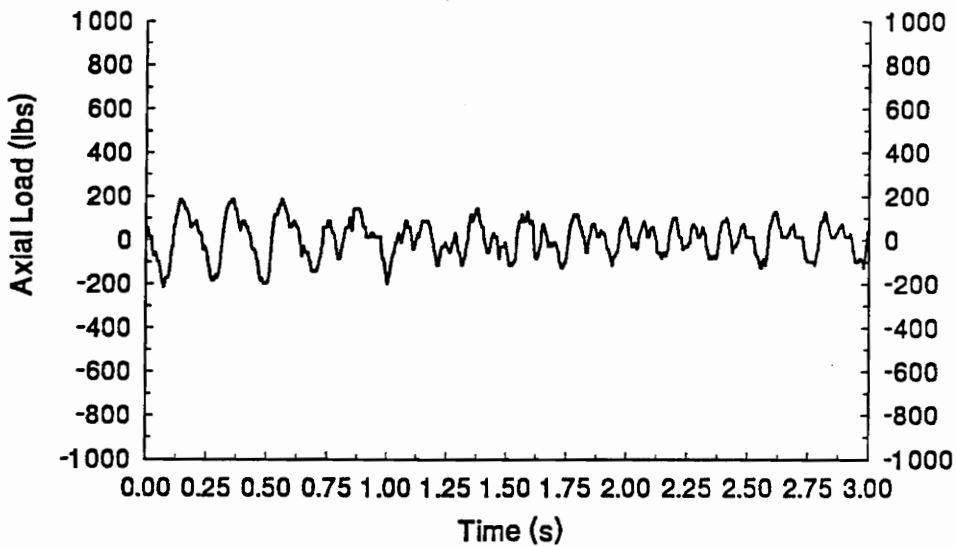


Figure 16. Typical Pitch Link Load History

To reduce loads while also retaining the proper geometry, internal flexibility and damping is designed into the main-rotor blades system. One critical component which allows flexibility and damping is the lead-lag damper, as seen in Figure 17. This simplified illustration shows a main-rotor blade pinned vertically to allow hunting. The two lead-lag dampers per blade (which are also loaded

axially) restrict and damp this motion. The range of loads on the monitored lead-lag damper was about - 1700 to 3700 lbf for the given maneuvers.

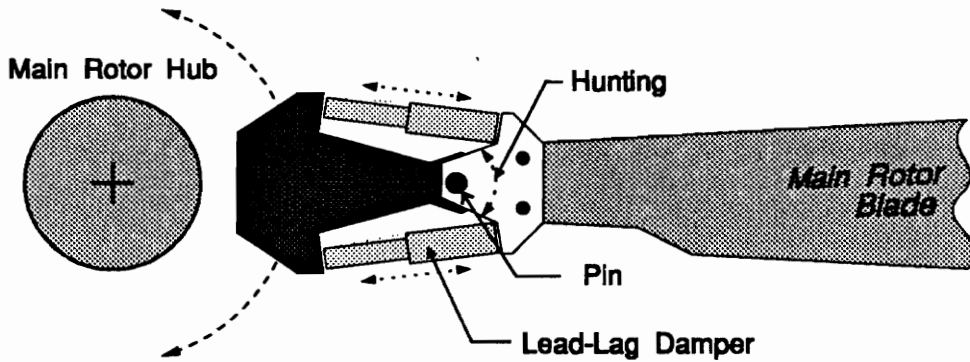


Figure 17. Axially Loaded Lead-Lag Damper

As with the other two critical components, the lead-lag damper is loaded both transiently and cyclically. Transient loads are attributed to collective changes in pitch which change the torque transmitted to the main-rotor shaft through the lead-lag dampers. Because the seven maneuvers included no changes in collective pitch, the transient constituent of the lead-lag damper load, similar to the pitch link load, is relatively small. This is demonstrated in Figure 18 which is a sample trace of the lead-lag damper load from one of the seven maneuvers of consideration. The cyclic constituents are again attributed to cyclic changes in the blade pitch and are multiples of one and four times the main-rotor frequency.

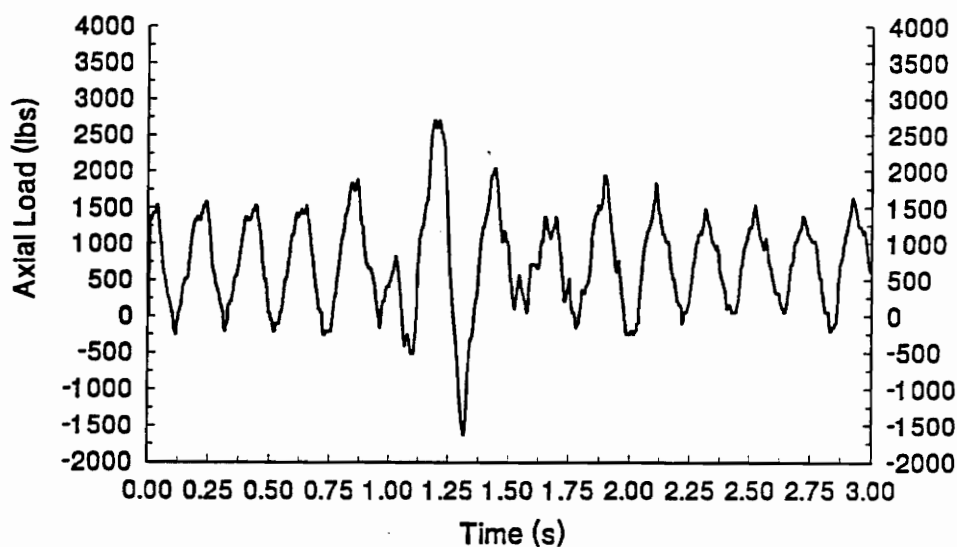


Figure 18. Typical Lead-Lag Damper Load History

In summary, the current neural network has been developed using data from seven maneuvers to model the nonlinear relationships between eight flight variables and three critical component loads: bending in the vertical plane of the tailboom, and axial loading of a lead-lag damper and a pitch link.

3.3 Data Management

Because neural networks learn to model the underlying relationships of a system through exposure to measured or desired behavior, there is inevitably a significant amount of data management involved. This process begins with data acquisition. If previous knowledge of the system permits, the task can be reduced to collecting data from only the pertinent variables. In other cases, data has already been collected and the task is reduced to sorting out the most useful information. The more difficult situation is one where data must be collected without any foreknowledge of which variables are pertinent. Additionally, care must be taken to gather data that will thoroughly describe the system. In general, it is said that the best data for training a neural network contains "rich" stimulus, or that which contains a

variety of both sinusoidal and non-sinusoidal inputs and responses [6]. The step following data acquisition is data conditioning, which involves signal processing for compatibility with a neural network.

3.3.1 Feature selection

The term "feature selection" can be clarified by considering the application of neural networks to image recognition. In these problems, certain features are more helpful in leading to the identification of an image. The objective of feature selection is to choose the most distinguishing features. In the helicopter loads application, important features are the available flight parameters which contribute to the resolution of a particular load.

Feature selection can be accomplished by several means. The most basic method is to select variables which are known to play important roles, either through experience or by engineering judgment. A more complicated method is statistical correlation. If variables are correlated with the output, there is a good chance that they will be useful to the network. However, this is not always the case because variables can be statistically correlated without being functions of each other [15].

A neural network method of selecting variables is to estimate the relevance of inputs by analyzing the converged network. One notion is that strong weights indicate level of relevance, but this also is not always the case. For example, sometimes hidden layer units become "constants" within the network meaning that they produce maximum or minimum output regardless of the stimulus. In these situations, the connection strengths to that unit are irrelevant. A more effective method is to test a trained network, quantifying its performance, and then set selected inputs to zero and test again on the same cases. The relevance of these parameters is the difference in performance with and without their input [15].

One other method is to "pick-and-train" until the best results are obtained. Although simplistic, this technique can be effective when the number of available parameters is relatively small. However, if there are, perhaps, fifty or more possible inputs for a twenty-input network, the computational energy

required to train and evaluate the performance of each subset of inputs is astounding. Also, the problem of feature selection is compounded when one considers creating additional inputs as functions and/or combinations of the initial inputs. In spite of these limitations, though, automated implementations of the pick-and-train method have been used successfully in recent applications [16],[17].

In the current application, the eight flight variables have been chosen based on knowledgeable selection, engineering judgement, and availability. These selections are only the beginning of the feature selection process for this application. Another method has been investigated and is nearly ready for implementation. In particular, a feature selecting algorithm, which tests a trained network with selected input parameters removed to assess their relevance, is currently being developed and will be implemented in the near future.

3.3.2 Data Conditioning

Almost every neural network application requires a significant amount of data conditioning. In particular, input and output data is typically scaled to match the neural network's "operating zone" which is normally the upper and lower limits of the squashing function. This step is necessary to make the parameters meaningful to the network and to initially assign uniform importance to each parameter. Scaling, or normalization, also helps to prevent processing units within the network from becoming "paralyzed". A unit becomes paralyzed when it is over- or understimulated during training because the derivative of the squashing functions approaches zero at the extremes, preventing training of the connecting weights (the updating of weights depends directly on the magnitude of the derivative at the stimulated level, as described in a latter section, 3.4.3). Also, it is common practice to use several neural networks or neural networks in conjunction with other techniques to solve a given problem which implies more pre- and postprocessing in combining input information and solutions. Thus, it is apparent that the use of a neural network in the modeling of any system will involve a significant amount of data management as well as signal processing.

In conditioning the helicopter input data for the neural network, the first step was normalization of both the input and output variables for the maneuvers under consideration. The most difficult part of this process was choosing effective normalizing factors for each variable. The data report, reference 9, gives operating limits for each variable, but examination of the data for the seven maneuvers of interest showed these limits to be extremely loose with respect to the moderate nature of the chosen maneuvers. Another alternative was to select minimums and maximums directly from the data set at hand. However, this would result in misrepresentation of some parameters. For example, a variable that had no meaningful variations would be scaled such that trivial variations become unrealistically magnified. Therefore, a compromise was made by using engineering judgement and neural network experience to choose factors that would scale the variables in a realistic and effective manner. For example, in the data report, the minimum and maximum vertical acceleration limits are listed as -2.2 and 5.0 G's. The use of these limits would nullify any useful information contained in the seven maneuvers. Instead, limits that are more relevant to the selected maneuvers were chosen as 0.5 and 1.5 G's. These boundaries allow the network to perceive small changes without necessarily oversizing the insignificant variations. Using this same philosophy, the normalizing factors for each of the eight input variables were chosen as shown in Table 3.

Table 3. Input Parameter (Flight Variable) Normalizing Factors

Parameter	Units	Min	Max
Pitch Rate	deg/sec	-25	25
Roll Rate	deg/sec	-50	50
Yaw Rate	deg/sec	-50	50
Vertical G's	G's	0.5	1.5
Lateral G's	G's	-0.5	0.5
Longitudinal G's	G's	-0.5	0.5
Longitudinal Control Position	%	50	80
Lateral Control Position	%	0	60

After choosing upper and lower bounds, each of the network input parameters was scaled according to the following equations:

$$X_{SCALED} = 0.5 + \left[\frac{X - offset}{range} \right], \quad (1)$$

where

$$offset = \frac{X_{MAX} + X_{MIN}}{2}, \quad (2)$$

and

$$range = X_{MAX} - X_{MIN}. \quad (3)$$

The three loads to be predicted were also normalized for use with the neural network. Table 4 shows the scaling factors selected for each load parameter. The output (load) parameters were also scaled according to equations (1) through (3), but the choice of normalization factors was accomplished with a different method. Before describing the method, however, some preliminary defining of the load predicting scheme is required.

Table 4. Output Parameter (Component Load) Normalizing Factors

Parameter	Units	Min	Max
Tailboom	in-lbf	-167,500	-100,000
Pitch Link	lbf	-150	150
Lead-Lag Damper	lbf	200	1800

The load-predicting network has been set up so that it predicts time-varying statistical variables rather than complete load histories. This step simplifies the prediction problem. More particularly, each of the three loads has been resolved into time-varying mean and oscillatory constituents. The calculation of these is fully described later. Currently, the network has only one output unit which predicts one constituent of the load on one helicopter component at a time, thus requiring a network with eight inputs

and only one output. To model the other constituent, the network is reconfigured with a new output variable. Therefore, to model both time-varying mean and oscillatory loads on three components requires six different network configurations.

In resolving each load into time-varying mean and oscillatory constituents, the first step was, again, normalization. The first "rule" adhered to in choosing scaling factors was that each critical load would be scaled only one time. In other words, even though each load is resolved into two constituents, only one pair of scaling factors would be used per load for all seven maneuvers. This was done to retain the relative magnitudes of the statistical variables with respect to each other for a given load. The second "rule" was that, after scaling and calculating the time-varying mean and oscillatory loads for a variable, the range of one of the new parameters over the maneuvers considered would be zero-to-one (the squashing function range) while the other parameter range would fall within zero-to-one. For example, after scaling and calculating the statistical parameters, the time-varying mean tailboom loads varied from zero-to-one. The range of time-varying oscillatory tailboom loads was a subset of the zero-to-one range. Implementing this rule required iteratively choosing factors, scaling the variables, and calculating and plotting the resulting constituents to see if the zero-to-one range was filled.

Utilizing the range of the network improves resolution of the predictions, similar to using the proper sized instrument when measuring some quantity. It can be viably argued that these narrow limits are not realistic when compared to the range of expected loads for the entire flight spectrum. But it is equally true that the resolution of current predictions is very high in comparison to the typical range of fatigue-damaging loads. Therefore, it is hoped that in expanding to a more extensive set of maneuvers, the resolution can be relaxed, compensating for the greater range of loads.

After normalization, the time-varying mean of each variable (input and output) is calculated with a sliding window. This process is analagous to high-frequency filtering or time averaging the data. Each of the seven maneuver records is about three seconds long, or 1400 points at a sample rate of 470

Hz. Calculation of the local mean is accomplished by averaging data in a 300 point window. This average becomes the time-varying mean for the center of that time slot, as shown in equation (4):

$$X_{MEAN}(n) = \frac{1}{300} \sum_{i=1}^{300} X_{SCALED}(n-150+i) \quad (4)$$

The window then slides forward one point to calculate the next average, and this process is repeated for cases $n=150 \dots 1250$. Because the sample rate is 470 Hz, this window corresponds to 0.64 seconds. Thus, the new signal is shortened by 0.32 seconds at the beginning and the end.

It is acknowledged that this choice of window size was not theoretically based. However, the 0.64 seconds is still meaningful. By experience, it was shown that smaller windows do not satisfactorily filter out the large 4.75 Hz sinusoidal component. On the other hand, the use of larger windows depletes too much data since the records are only three seconds long. Therefore, the 300 point window is a compromise. This concludes the statistical manipulation of the (input) flight variables. These scaled, time-averaged flight variables are the conditioned input variables for the neural network.

As mentioned earlier, the output variables are more extensively conditioned. The time-varying mean constituents are the scaled, time-averaged, or filtered signals which are calculated as described above. But the time-varying oscillatory loads require further calculations. After scaling and calculating the mean constituent, the time-varying standard deviation is calculated and called the time-varying oscillatory load. This parameter represents the magnitude of the chief sinusoidal component in the data. The time-varying oscillatory constituent is simply the time-averaged standard deviation, as shown in equation (5), calculated also with a sliding window of 300 data points.

$$X_{OSCI}(n) = \sqrt{\frac{1}{299} \left(\sum_{i=1}^{300} [X_{SCALED}(n-150+i) - X_{MEAN}(n-150+i)]^2 \right)} \quad (5)$$

Here, the calculation is repeated for cases $n=300 \dots 1100$. Therefore, the final conditioned data is a total of 1.28 seconds shorter than the original signal.

The final step in manipulating the data is randomization for training. The values of each of the eight conditioned flight variables and the load constituent corresponding to an instant in time constitute one training case. These cases are sequentially ordered row by row in a file for one maneuver. This original order is used for testing a trained network. However, for training purposes, it is necessary to randomize the order which the data is presented to the network. This prevents the network from converging to a local solution which may occur when several similar cases are presented sequentially. Randomization is accomplished by generating random numbers and assigning one to each case of data. The cases are then reordered according to their associated random numbers and saved in a training file.

Admittedly, there several more steps involved in preparing the data for use with the neural network. Most of these involve data management, such as combining files for different training and testing configurations. The demonstration of the load-predicting neural network, Chapter 4, includes more descriptions of how the files are combined and presented to the network in the different phases of training and testing.

3.4 Load-predicting Neural Network

This section begins by describing the general load-predicting neural network. As stated earlier, this network predicts either the time-varying mean or oscillatory constituent of the load on one component at a time. This simplifies the problem by dividing it into several parts. The different components of each load are considered separate modeling problems for the neural network. The current neural network, a feedforward multilayer type, is trained with the popular back-propagation algorithm. This choice of network type and training algorithm was based on 1) the history of successful applications and 2) the availability of references and resources through professional associates. The general configuration of the load-predicting neural network that has proven most successful is given in Figure 19.

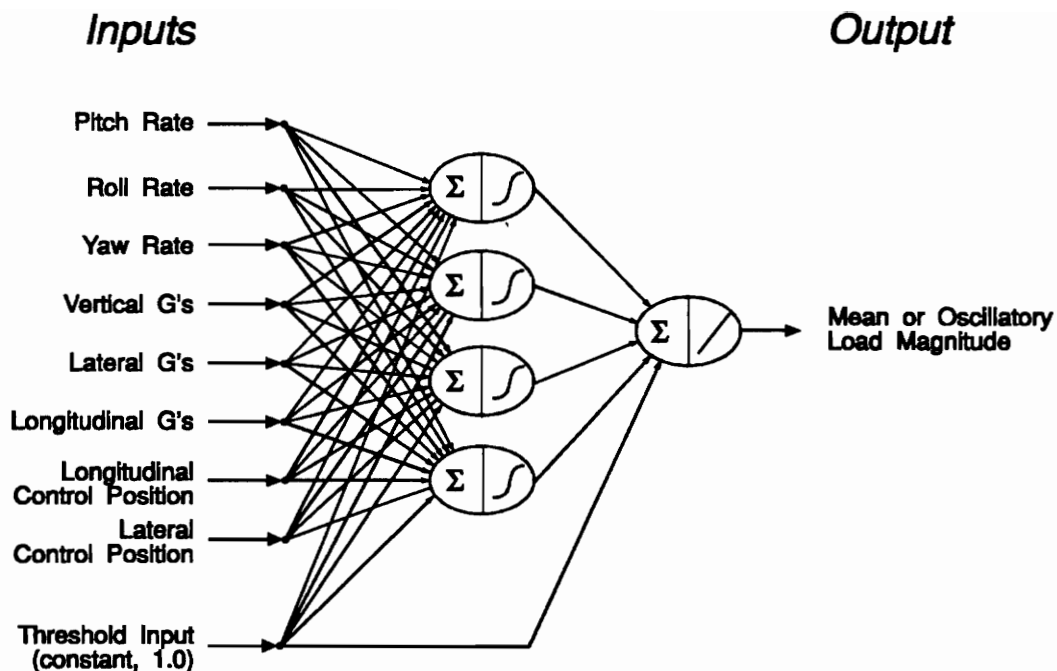


Figure 19. Load-predicting Neural Network

As shown in the figure, this network has three layers; an input layer of eight flight variables and a threshold (this input provides a weighted bias for each processing unit to operate about), a hidden layer consisting of a variable number of units with sigmoid squashing functions, and an output layer of only one unit with a linear function. The network is "fully connected", meaning that each unit in a given layer is connected to every unit in the following layer, as shown in the illustration. Connections, or weights, can be positive or negative, causing excitatory or inhibitory stimulus to the receiving unit.

The nonlinear function used in the hidden layer units is the sigmoid squashing type. This particular sigmoid function approaches zero for very negative input and plus-one for large positive input. The linear function of the output unit facilitates the prediction of values throughout the range from zero to one. This is different from sigmoid squashing functions which, as output units, are better suited for classification type problems where the desired output patterns are binary. Both functions used in the current neural network satisfy the requirement of a continuous derivative over the entire range of

stimulus. The derivative is used in calculating weight updates during training. For load predicting, the output of the network indicates the instantaneous normalized magnitude of either the time-varying mean or oscillatory load being considered.

The network is trained with a slight variation of the popular back-propagation technique (subsequently described) which seeks to minimize the least-mean-square (LMS) error. A tolerance term is included in training which allows the algorithm to ignore trivial cases. This is intended to reduce training time and allow "flexibility" in the network so that the more difficult boundary cases can be better modeled.

The operation of the neural network is described in four basic modes. To begin, the network is initialized by choosing geometry and training parameters. The next step is feedforward operation. Here, the network receives stimulus or a vector of input parameters and the information is operated on by the connections and processing units as it feeds forward through the network. In the training phase, the output is compared to the correct output and this error is attributed to individual connections in the network in a back-propagating fashion. When the network is finally trained, the weights are fixed and kept for testing. In testing, the network operates in the feedforward mode to produce outputs which can be compared to the proper output to quantify performance. Here, no adjustment of the internal weights takes place. The following sections describe the analytical details of these four modes of operation for the load-predicting neural network.

3.4.1 Neural Network Initialization

Initialization of the network begins with specification of network geometry, including the number of layers and number of units in each layer. In addition, learning parameters such as training sensitivity, or gain, and tolerance are specified. Finally, the interconnections are given small initial random values. In the present application, the training sensitivity, η , is usually set to 0.40 for both layers and the initial weights are randomly distributed between plus and minus 0.30. Most of the guidelines for setting these variables are rule-of-thumb, and/or derived from individual experience.

3.4.2 Feedforward Operation

As mentioned earlier, the operation of a neural network is based on the biological understanding of the human brain. A large number of simple processing units, analogous to neurons in the brain, are densely interconnected by weights, or synapses in the brain, and operate in a parallel fashion. Neural network processing units produce a nonlinear output based on the sum of their input values, similar to neurons in the brain which "fire" an electrical impulse when they receive sufficient stimulus. Typical nonlinear functions are said to "squash" the sum of the inputs into the output range of the unit. The output of the unit is then modified and transmitted to other units via connections, the same way neuron output is transmitted to other neurons via synapses. To implement these ideas, neural networks are organized into layers of simple processing units operating in parallel, connected to all units in adjacent layers, and passing information in a feedforward fashion. Typical neural networks consist of three or four layers; input and output layers plus one or two middle layers, often referred to as hidden layers. In particular, the load-predicting neural network operates in the feedforward mode as described in the following paragraphs and illustrated in Figure 20. For clarity in presenting the equations, the separate functions of summing and squashing are shown separately for each processing unit, even though both are performed by a single unit. Also, the threshold inputs are shaded and shown separately for the hidden layer and output layer.

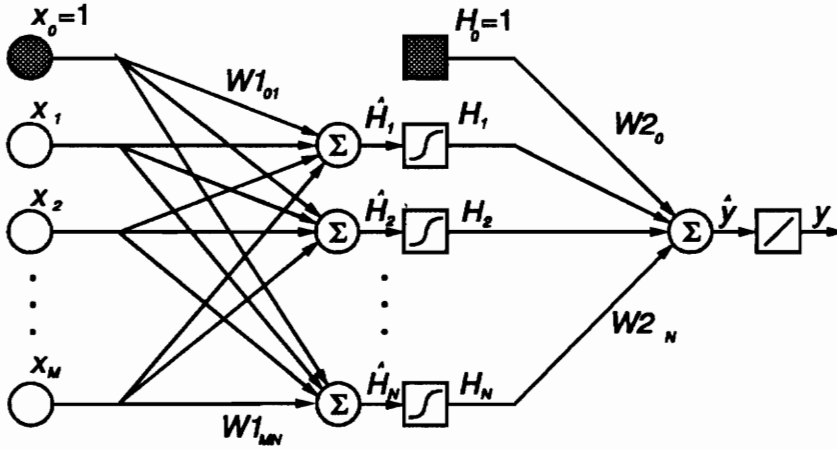


Figure 20. Load-predicting Neural Network Operating Schematic

The conditioned values of the eight flight variables from a specific instant in time and the threshold, x_0 , become the input vector. Each value in the vector is weighted and transmitted by the connections to each unit in the hidden layer. Hidden layer units sum the weighted inputs plus a weighted threshold as shown in equation (6).

$$\hat{H}_n = \sum_{m=0}^M x_m W1_{mn} \quad (6)$$

Here, x is the current value of input m and $W1$ is the weighted connection from input m to hidden layer unit n . Hidden layer units squash their sum, \hat{H} , to an output between zero and one according to equation (7):

$$H_n = \frac{1}{1 + \exp^{-\hat{H}_n}} \quad (7)$$

This is the sigmoid squashing function.

The output, H , of unit n is then weighted and transmitted to the output unit y . This single output unit also sums its weighted inputs plus a threshold as in equation (8);

$$\hat{y} = \sum_{n=0}^N H_n W2_n, \quad (8)$$

where $W2$ is the weight from hidden layer unit n to the output unit. The network output is simply one-quarter of the sum:

$$y = 0.25\hat{y} \quad (9)$$

The factor 0.25 gives this linear function the same output and slope as the sigmoid function for a null input. Thus, the single output of the network is a linear function of the weighted sum from the hidden layer unit outputs which are nonlinear functions of the weighted inputs.

3.4.3 Back-propagation Training

Training of the neural network is more complicated than feedforward operation. In fact, the success of the neural network idea was delayed for a period of time due to the lack of an effective training algorithm, in particular, for adjusting weights in the hidden layers. Finally, the back-propagation algorithm was developed [7].

This algorithm is a generalization of the least-mean-square (LMS) error algorithm, which seeks to minimize the average squared error for all outputs. Similar to LMS, the back-propagation algorithm uses a gradient search technique to adapt the internal weights, minimizing the cost function. Here, the error of an output unit is calculated and attributed to weights in the preceding layers. These weights are then adapted based upon their stimulus, the error at the output unit, and the slope of the output unit's squashing function at its level of stimulus. Next, the error is propagated to weights in the preceding layers and they are adapted in a similar manner [4]. With this in mind, the load-predicting back-propagation equations are derived next.

The LMS error is defined for a network with several outputs as follows:

$$E_{LMS} = \frac{1}{2} \sum_{j=1}^J (d_j - y_j)^2 \quad (10)$$

where d and y are the desired and network outputs respectively for output unit j . For the load-predicting network, there is only one output ($J=1$), but the same definition of error is used in deriving the training equations. To minimize this error, the weights should be changed opposite and in proportion to the direction and magnitude of the derivative of the error with respect to the weights. Beginning with the second layer of weights,

$$\Delta W_{2_n} \propto -\frac{\partial E_{LMS}}{\partial W_{2_n}} \quad (11)$$

This is expanded and simplified with the chain rule;

$$-\frac{\partial E_{LMS}}{\partial W_{2_n}} = -\frac{\partial E_{LMS}}{\partial y} \frac{\partial y}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial W_{2_n}}, \quad (12)$$

where

$$\frac{-\partial E_{LMS}}{\partial y} = d - y,$$

$$\frac{\partial y}{\partial \hat{y}} = 0.25,$$

and

$$\frac{\partial \hat{y}}{\partial W_{2_n}} = H_n.$$

Thus, the weight changes for the second layer should be

$$\Delta W_{2_n} \propto -\frac{\partial E_{LMS}}{\partial W_{2_n}} = 0.25(d - y)H_n \quad (13)$$

This result is implemented with the addition of a gain term (or learning sensitivity), η , which controls the rate of change in updating weights:

$$W2_{n_{mv}} = W2_{n_{sz}} + \eta(0.25)(d - y)H_n, \quad (14)$$

$$(0 < \eta < 1).$$

Adjustment of the first layer of weights is derived in a similar fashion. Again, weight changes should be opposite and in proportion to the direction and magnitude of the derivative of the LMS error with respect to the weights as in equation (15).

$$\Delta W1_{mn} \propto -\frac{\partial E_{LMS}}{\partial W1_{mn}}. \quad (15)$$

This expression is also expanded and simplified with the chain rule below.

$$-\frac{\partial E_{LMS}}{\partial W1_{mn}} = -\frac{\partial E_{LMS}}{\partial H_n} \frac{\partial H_n}{\partial \hat{H}_n} \frac{\partial \hat{H}_n}{\partial W1_{mn}}, \quad (16)$$

where

$$-\frac{\partial E_{LMS}}{\partial H_n} = -\frac{\partial E_{LMS}}{\partial y} \frac{\partial y}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial H_n},$$

$$= (d - y)(0.25)W2_n,$$

$$\frac{\partial H_n}{\partial \hat{H}_n} = \frac{\partial}{\partial \hat{H}_n} \left[\frac{1}{1 + \exp^{-\hat{H}_n}} \right],$$

$$= \frac{1}{1 + \exp^{-\hat{H}_n}} \left[1 - \frac{1}{1 + \exp^{-\hat{H}_n}} \right],$$

$$= H_n(1 - H_n).$$

[see equation (7)]

And

$$\frac{\partial \hat{H}_n}{\partial W1_{mn}} = x_m$$

Combining and regrouping yields

$$\Delta W1_{mn} \propto -\frac{\partial E_{LMS}}{\partial W1_{mn}} = 0.25(d-y)W2_n[H_n(1-H_n)]x_m \quad (17)$$

This result is implemented with the addition of the gain term, η , similar to equation (14):

$$W1_{mn_{new}} = W1_{mn_{old}} + \eta(0.25)(d-y)W2_n[H_n(1-H_n)]x_m, \quad (18)$$

$$(0 < \eta < 1)$$

The resulting equations, (14) and (18), are used in training as follows. After initializing the network, an input vector is presented along with the desired output. The network output is then calculated in the feedforward mode and compared with the desired output to obtain an error. This error is compared to the prescribed tolerance which, if met, prevents any weight changes. When the tolerance is not met, the error is used first to update the second layer of connecting weights according to equation (14), and then to adjust the first layer of weights according to equation (18). Next, a new vector of inputs and the corresponding output is presented to the network and the process begins again. When all cases in a training set have been presented, one iteration has occurred. The training data is presented repeatedly for many iterations (sometimes thousands) until the error is reduced to an acceptable level for the entire set. When this occurs, the network has converged to a solution and the weights are saved for testing.

3.4.4 Analysis of Network Performance

Network training ends when the network either performs satisfactorily or fails to converge. At this point, the network performance can be quantified with test cases. The load-predicting network is tested both on training data and separate test data. In both situations, the cases are presented in time-sequential order so that the results can be plotted. In testing, the network operates in the feedforward mode to produce modeled response for comparison to known system output. Because the output range for the load-predicting network is zero-to-one, a measure of the percent-of-scale error for case k is obtained by

$$E(k) = 100[d(k) - y(k)] \quad (19)$$

This quantity can be averaged over the set of K cases for an overall measure of performance as in equation (20):

$$\bar{E} = \frac{100}{K} \sum_{k=1}^K [d(k) - y(k)] \quad (20)$$

In summary, section 3.4 details the operation of the load-predicting neural network. After training, the network is tested both on training and non-training data and the results plotted. A quantification of performance is also established.

4 Demonstration of Load-predicting Neural Network

As mentioned earlier, neural networks are useful for modeling complicated nonlinear relationships. This thesis demonstrates this through the application of neural networks to indirect helicopter load monitoring from flight variables. The following sections give an overview of the load-predicting neural network's performance. Loads on three components have been predicted: one in the fixed system and two in the rotating system. The envelope of data used for this demonstration contains seven maneuvers. For simplicity, only the results from two maneuvers are presented. These results verify progress towards the development of neural network tools for this specific application and justify pursuing this method of relating critical component loads to flight variables.

4.1 Training and Testing Procedure

Section 3.2.2 described how the original flight variable and component load histories are conditioned for use with the neural network, including normalization, time-averaging for the input variables, and calculation of the time-varying mean or oscillatory load constituent (output variable). This section is a continued description of how the data is prepared with regard to combining data from different maneuvers for training and testing purposes.

The general process for training and testing the load-predicting network is shown schematically in Figure 21. This flow chart shows the five steps necessary to predict one constituent (mean or

oscillatory) of one of the three critical component loads being considered. To predict the other loads, new testing and training files must be created and the network retrained.

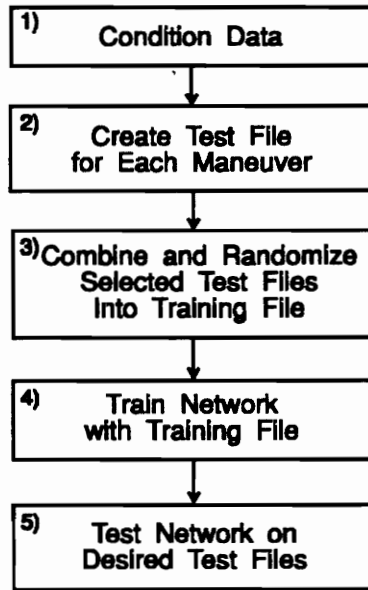


Figure 21. Neural Network Training and Testing Procedure

After the variables from one maneuver have been conditioned (step (1)), as in section 3.3.2, they are combined into a matrix where each row represents one case of data (the conditioned values of all nine parameters for a specific instant in time). At this point, the cases are still sequentially ordered. To conserve computation time and disk space, every fourth row, or case, is then saved in a file for testing. It was determined that this had no visible effect on the plotted results when testing. This procedure is then repeated to create individual test files for each of the seven maneuvers, completing step (2).

Step (3), in accordance with Figure 21, is generating training data. This is begun by selecting maneuvers to be used for training (the method of choosing training and testing combinations is addressed in the following section). To generate a training file, test files corresponding to the selected maneuvers for training are patched together end-to-end in a matrix. Each case of data in this file is assigned a

random number. The cases are then reordered according to their associated random numbers and saved as a training file. Thus, training files are randomly ordered cases of data taken directly from test files.

Once the desired training file has been created, the network is finally ready to train, step (4). The training algorithm is implemented with a FORTRAN program which requires a training file and the user's specification of training instructions. These include the number of hidden layer units, the number of iterations to complete, the training tolerance, and the learning rate μ . When training is complete, the most recent interconnecting weight values are stored for testing purposes. These weights fully describe the converged network.

Step (5), according to Figure 21, consists of testing the converged network on the desired maneuvers. Testing of a trained network is accomplished by presenting time sequential cases of input data from a test file and comparing the predicted output with known output. This procedure is also executed with a FORTRAN program implementing only the feedforward mode of network operation.

This concludes the process necessary to use the neural network to predict one load constituent. In order to predict other load constituents, the five steps shown in Figure 21 are repeated with a new output variable.

4.2 Results

Before presenting the network prediction results, one final note about training is necessary. The following results represent repeated executions of the previous training and testing procedures. As mentioned, the user must assign values to training specifications such as the learning rate, number of hidden layer units, amount of tolerance, and number of training iterations (passes through the training data). Preferable selections are those that lead to accurate modeling over a variety of test conditions without excessively long training times. However, guidelines for making these choices are quite sparse in the current state-of-the art literature [6]. Therefore, experience in applying the network to the system of interest becomes the best reference.

In training the load-predicting neural network, a range of favorable training specifications was established. Learning rates ranged from 0.3 to 0.6, but were typically 0.4. The number of training iterations ranged from 300 to 1000 with 500 being typical. Tolerances ranged from 0 to 4%, with 2% being the norm. Most of the predictions were generated with only four units in the hidden layer, but a six unit hidden layer was also used.

The effects of variations within these ranges on the trained network performance were found to be trivial. It was evident in each case that, regardless of the specifications, the network had reached a performance limit when training was complete. Therefore, further detail about the specifications for each training case is omitted. A more detailed discussion about other effects of changing these variables follows the presentation of the test results.

Test results from different maneuvers can be divided into two groups; testing on data that was, or was not, included in training. Thus, the type of result depends on which maneuvers the network was trained on and which maneuvers it was tested on. For the following demonstration of the load-predicting neural network, two types of training files have been used; The first type contained the conditioned, randomized data from all seven maneuvers. After training, the converged network was tested on each of the seven maneuvers individually. Hence, the test data was included in training. For clarity in the rest of the document, this method of analyzing the network will be referred to as "mode-one". The philosophy for this mode of testing is that, ideally, in the ultimate realization of indirect load monitoring with neural networks, all possible combinations of maneuvers (i.e: the entire flight spectrum) can be included in the network training database. The second type of training files consisted of data from only six maneuvers. The network was trained with these six maneuvers and then tested on the seventh. Hence, for the second mode of analysis, "mode-two", the test data was not included in training. To obtain type-two results for the other six maneuvers, new training files were generated and the network was trained and tested on the odd maneuver. Mode-two results more effectively analyze the network's ability to generalize, and demonstrate an understanding of the underlying relationships rather than memorization of a few

maneuvers. Thus, the two modes of analysis, testing on training data (or maneuvers) and testing on non-training maneuvers, allow different aspects of neural network performance to be observed and studied.

As stated in the previous paragraph, the load-predicting network has been analyzed in two modes: 1) training with all seven maneuvers and testing on the same maneuvers individually, and 2) training with six maneuvers and testing on the seventh. Given that there are three critical components, two statistical load constituents per component, two modes of analysis, and seven maneuvers, the total number of plotted results would be eight-four! With this in mind, only representative results from two maneuvers are presented.

As shown in Figure 22, an "easy" and a "difficult" maneuver have been selected for presentation. The designations of "easy" and "difficult" are based upon maneuver location in X-Y stick position coordinate space, and refer to the level of anticipated difficulty for neural network predictions. The $-0.5X$ maneuver is presumed to be an easy case (relative to the others) because it is bounded by $-1.0X$ and $+1.0X$, and it is mirrored by the maneuver $+0.5X$. The $+1.0Y$ maneuver is presumed to be difficult because it is a boundary case, and there is less data in the Y direction than X. This maneuver is particularly difficult when testing in mode-two (the network is trained without that maneuver) because the network is required to extrapolate from the training data. This is termed extrapolation because the training maneuvers for that mode range from -1.0 to $+0.5$, in the Y direction. Thus, these two maneuvers represent both directions (X and Y), both magnitudes (0.5 and 1.0), and two levels of difficulty.

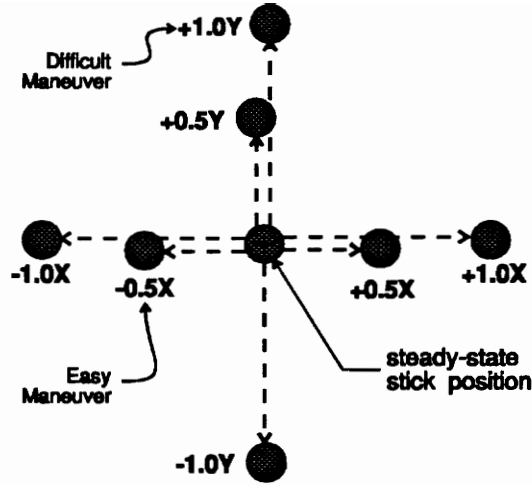


Figure 22. "Easy" and "Difficult" Maneuvers Selected for Presentation

Figures 23 and 24 show the eight conditioned flight variable histories corresponding to the two maneuvers selected for presentation. As seen in the plots, most of these input variables remained fairly constant during the maneuvers, with the exception of a few. For the $-0.5X$ maneuver, the variables that show significant variations are the lateral stick control position and the roll rate. This makes sense because a change in lateral stick position should cause the aircraft to roll, and also there appears to be a time-lag between the two which reflects the aircraft's response time. There is also a small amount of variation in vertical G's. For the $+1.0Y$ maneuver, the variables that show significant variations are the longitudinal stick control position and the pitch rate. These also are to be expected, along with the small variations in vertical G's. The difference in severity of the two maneuvers is evidenced by the larger variations seen in the $+1.0Y$ flight variable histories. These conditioned flight records contain all the information that the neural network receives when tested over these maneuvers.

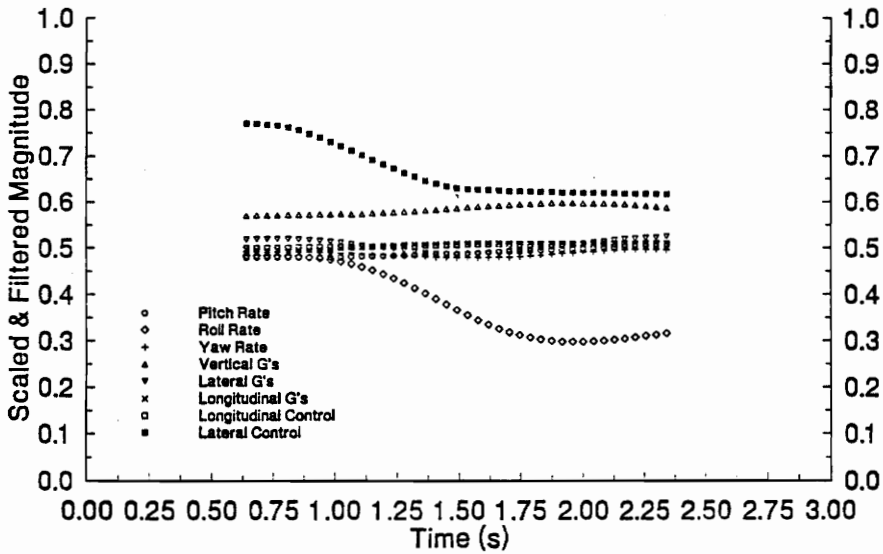


Figure 23. Conditioned Flight Variable Histories for -0.5X "Easy" Maneuver

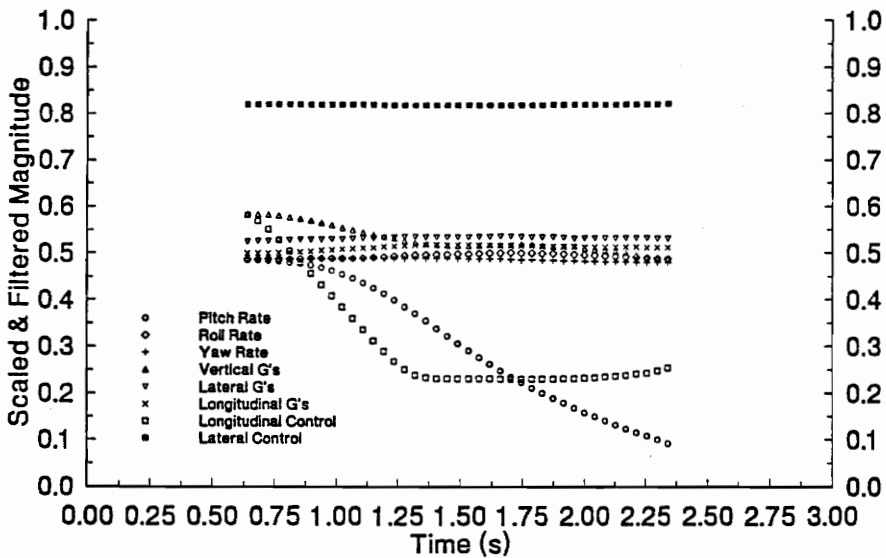


Figure 24. Conditioned Flight Variable Histories for +1.0Y "Difficult" Maneuver

Table 5 serves to further illuminate the spectrum of results to be presented. This table lists the component of interest, the maneuvers used to train the network, the test maneuver, and the corresponding

figure number. For clarity, the mean load and oscillatory load predictions for individual maneuvers are presented in the same figure, along with a plot of the original load history.

Table 5. Description of Presented Results

Loaded Component	Training Maneuvers	Test Maneuver	Figure Number
Tailboom	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	25
Tailboom	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	+1.0Y	26
Tailboom	+0.5X, +1.0X, ____, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	27
Tailboom	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, ____, -1.0Y	+1.0Y	28
Pitch Link	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	29
Pitch Link	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	+1.0Y	30
Pitch Link	+0.5X, +1.0X, ____, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	31
Pitch Link	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, ____, -1.0Y	+1.0Y	32
Lead-Lag Damper	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	33
Lead-Lag Damper	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	+1.0Y	34
Lead-Lag Damper	+0.5X, +1.0X, ____, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	35
Lead-Lag Damper	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, ____, -1.0Y	+1.0Y	36

To summarize and look ahead, the following sections contain prediction results and discussions for both mean and oscillatory load constituents on three components: the tailboom, a pitch link, and a lead-lag damper. Network performance has been analyzed in two modes: 1) testing on training data, and 2) testing on non-training data. After training, the converged network is presented with time-sequential vectors of input variables. The network produces an output time history which can be compared to the actual load constituent history. The plotted results from two representative maneuvers have been selected for presentation. Plots of the input variable records for these two maneuvers were shown previously in Figures 23 and 24. A discussion section follows the presentation of results.

4.2.1 Tailboom Results

Table 6 lists the training maneuvers and test maneuver for each figure, along with the corresponding average test error. The prediction results are on the order of 90 to 100% accurate. The first two rows correspond to mode-one analysis. Here the network was tested on maneuvers that were included in the training data. Because the training maneuvers were the same, the results presented in these two rows required only two networks: one for the mean predictions and one for the oscillatory predictions. In other words, the network was trained with all seven maneuvers to predict time-varying mean loads and tested over two maneuvers, and then trained to predict time-varying oscillatory loads and tested over two maneuvers. The last two rows correspond to mode-two analysis where the network was trained on six maneuvers and tested on the seventh. These results required training and testing of four individual networks.

Table 6. Tailboom Test Results

Figure Number	Training Maneuvers	Test Maneuver	Avg Error (%) Mean Predictions	Avg Error (%) Oscillatory Predictions
25	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	2.3	1.1
26	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	+1.0Y	2.2	3.6
27	+0.5X, +1.0X, _____, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	6.9	2.3
28	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, _____, -1.0Y	+1.0Y	9.3	5.2

The first and third rows correspond to testing on the "easy" maneuver (-0.5X) and the second and fourth rows correspond to testing on the "difficult" maneuver (+1.0Y). The average percent-of-scale errors shown in the table were calculated in accordance with equation (21), section 3.4.

Plots of the tailboom results appear in Figures 25 thru 28, as listed in Table 6. The top plot in each figure is the original load history. The middle plot is the actual and predicted time-varying mean load. The lower plot is the actual and predicted time-varying oscillatory load. Therefore, each figure contains the original load history and two sets of results.

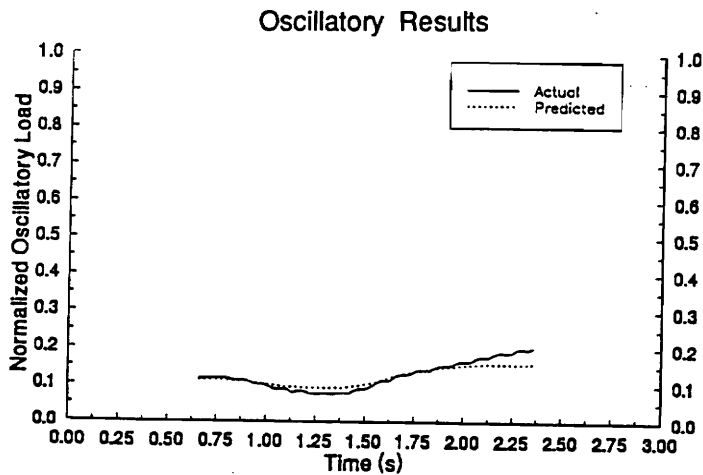
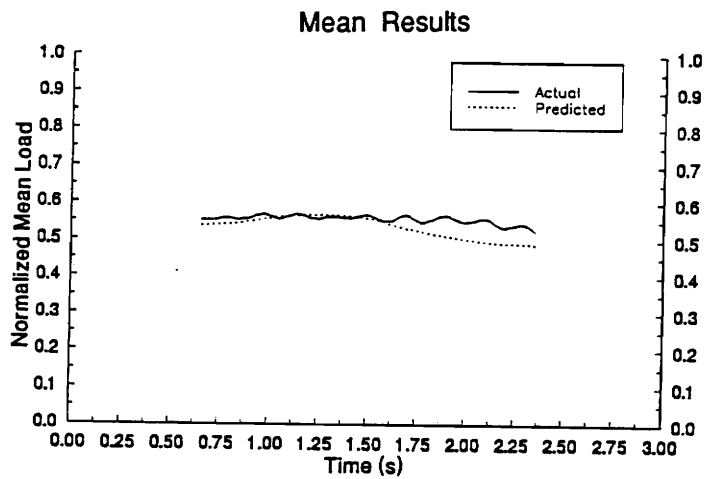
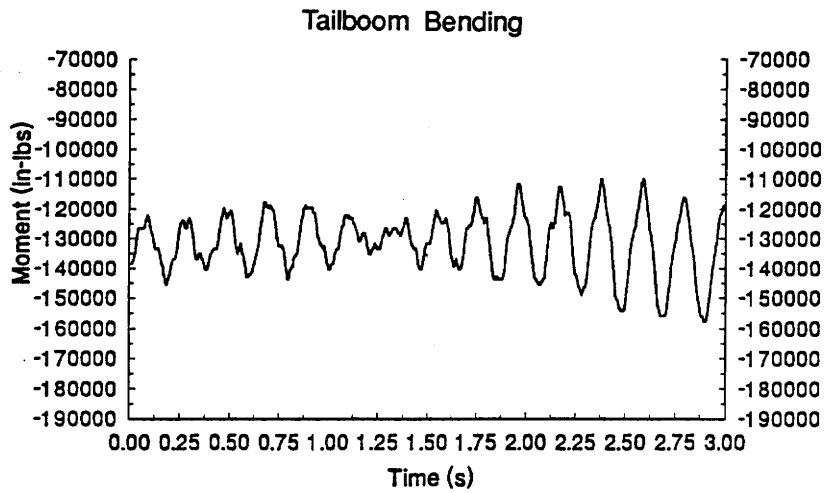


Figure 25. Tailboom Results. Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "easy")

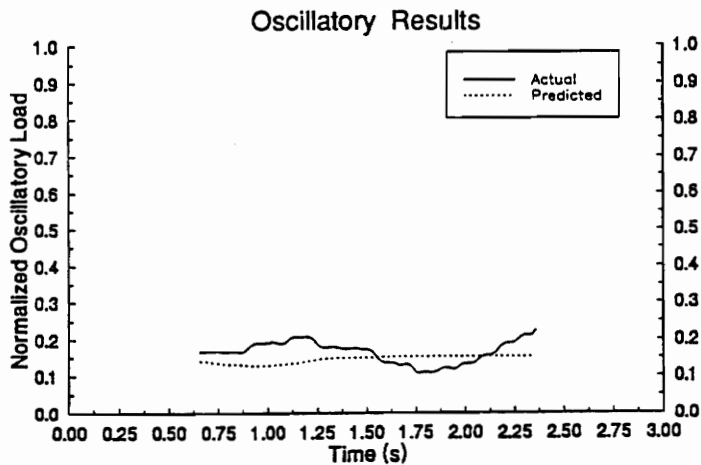
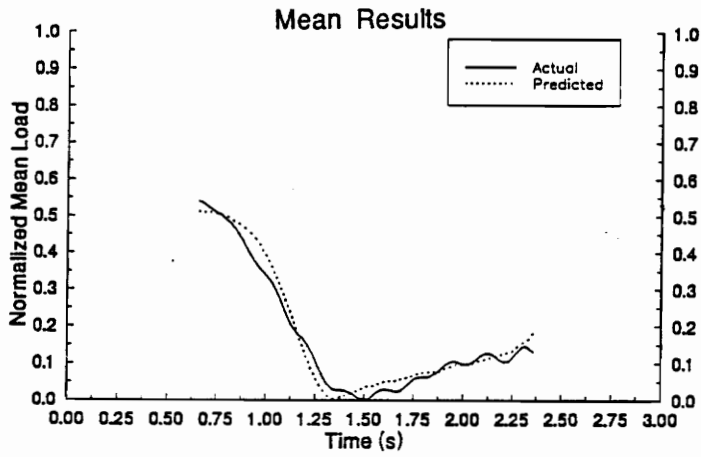
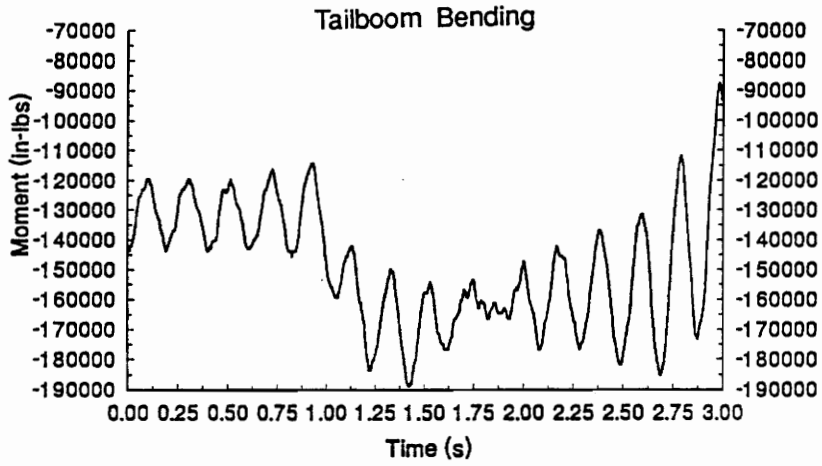


Figure 26. Tailboom Results. Test Maneuver: +1.0Y. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "difficult")

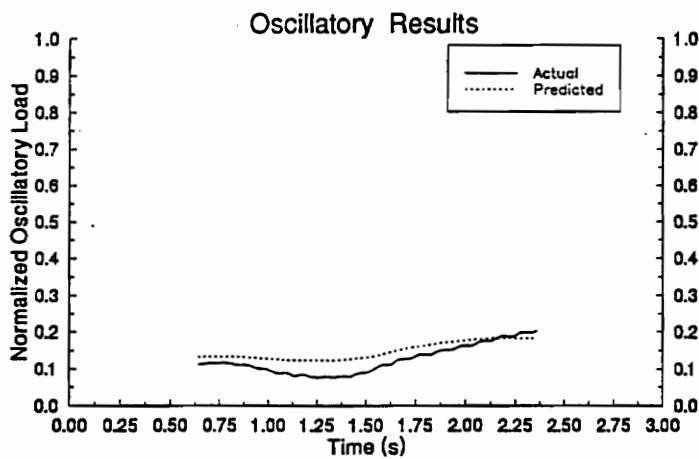
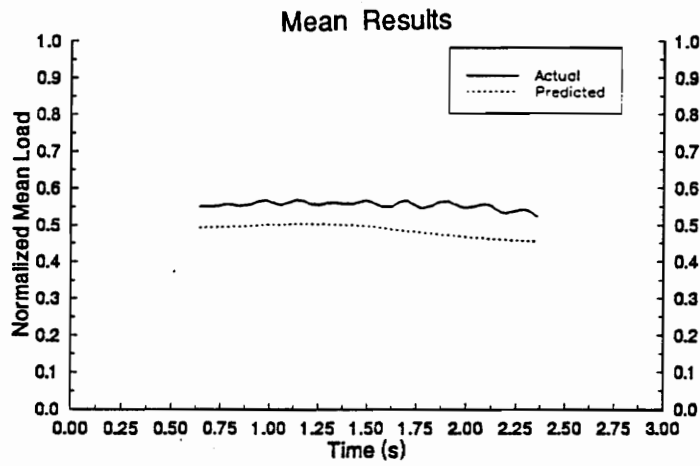
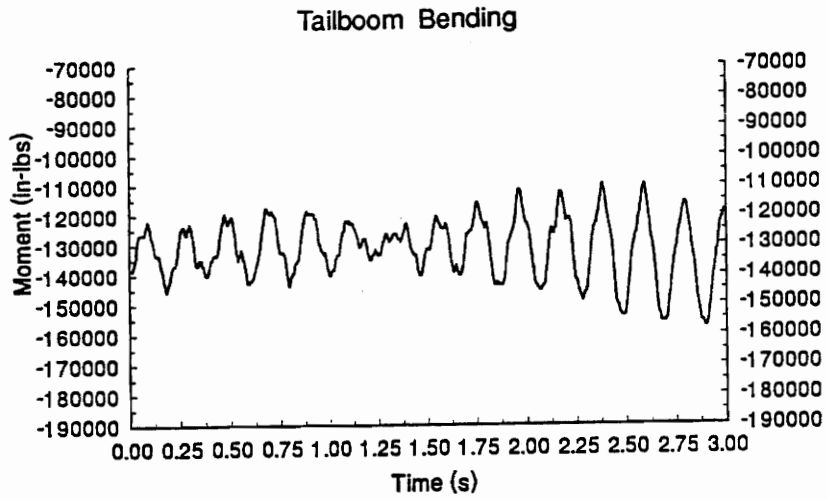


Figure 27. Tailboom Results. Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, ____, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-two, "easy")

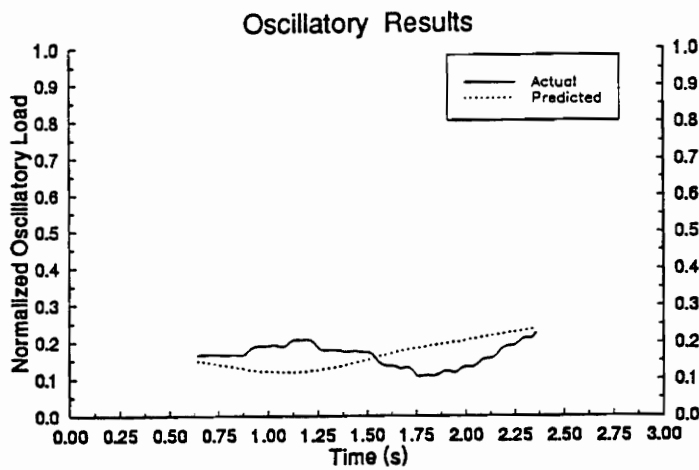
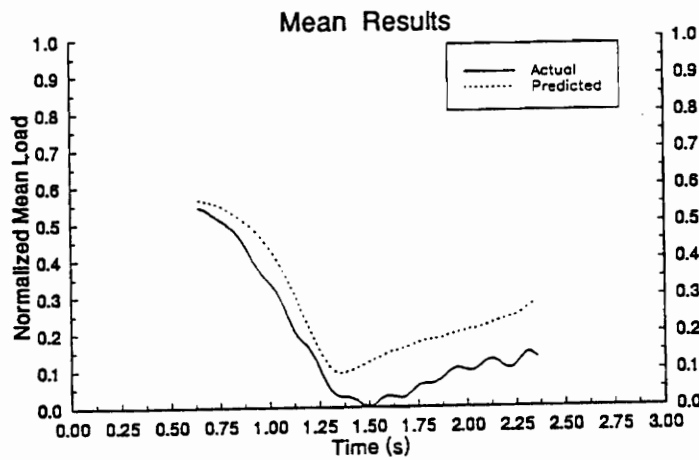
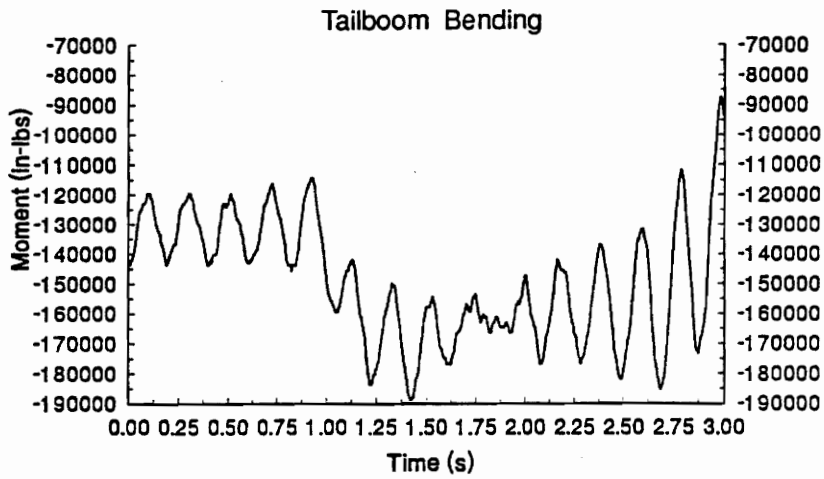


Figure 28. Tailboom Results: Test Maneuver, +1.0Y, Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, ____, -1.0Y (mode-two, "difficult")

4.2.2 Pitch Link Results

Table 7 shows the average percent-of-scale errors corresponding to the four training and testing combinations for the pitch link. The network predictions for the pitch link loads are on the order of 95% accurate. Again, the first two rows correspond to mode-one analysis. Here the networks were trained on all seven maneuvers and then tested on the two maneuvers individually. The last two rows correspond to mode-two analysis where the network was trained on six maneuvers and tested on the seventh.

Table 7. Pitch Link Test Results

Figure Number	Training Maneuvers	Test Maneuver	Avg Error (%) Mean Predictions	Avg Error (%) Oscillatory Predictions
29	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	4.4	1.8
30	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	+1.0Y	3.1	2.8
31	+0.5X, +1.0X, _____, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	4.3	3.2
32	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, _____, -1.0Y	+1.0Y	6.5	6.8

The first and third rows correspond to testing on the "easy" maneuver (-0.5X) and the second and fourth rows correspond to testing on the "difficult" maneuver (+1.0Y). The average percent-of-scale errors shown in this table were also calculated in accordance with equation (22), section 3.4.

Plots of the pitch link results appear in Figures 29 thru 32, as listed in Table 7. Like the tailboom results, each figure contains the original load history and two sets of results.

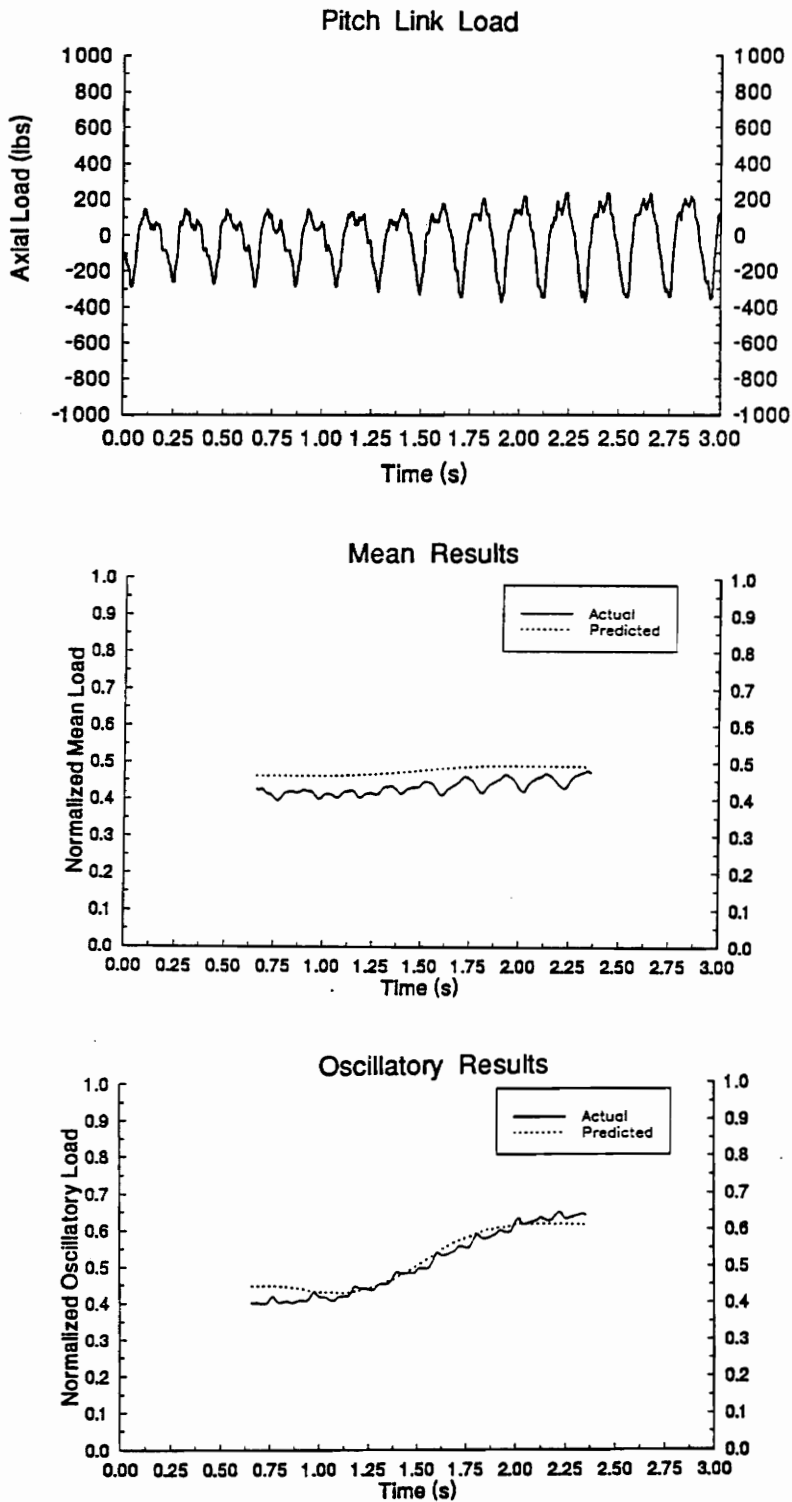


Figure 29. Pitch Link Results: Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "easy")

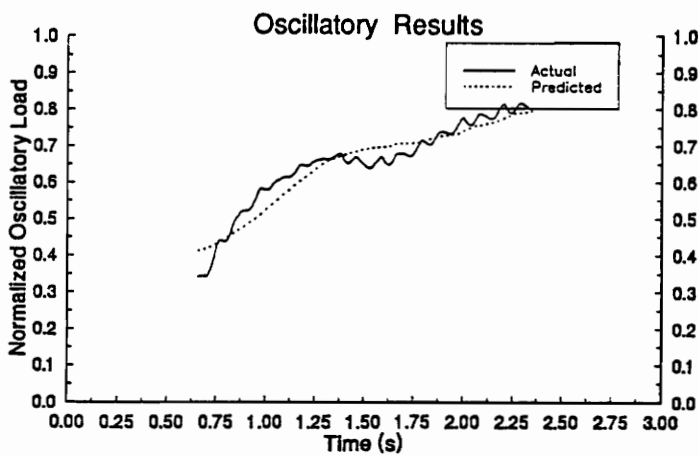
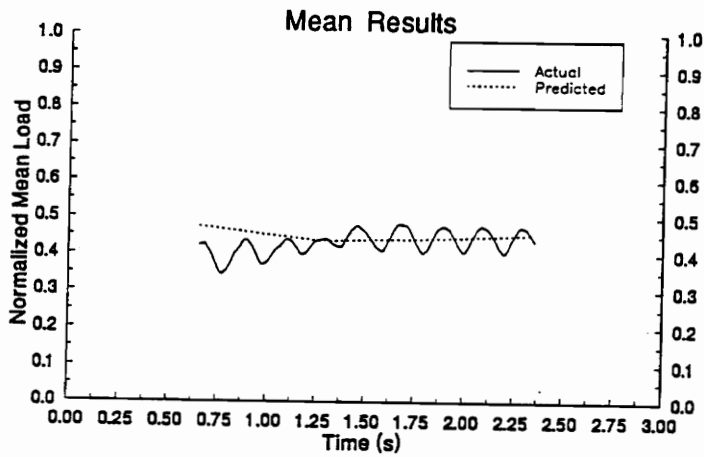
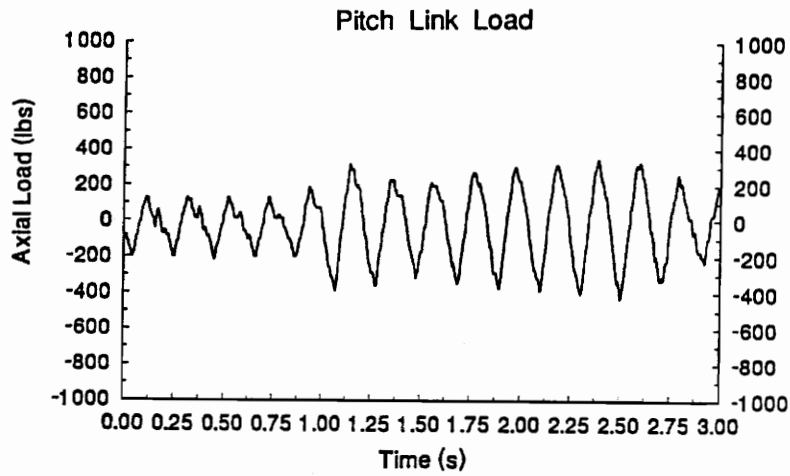


Figure 30. Pitch Link Results: Test Maneuver: +1.0Y. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "difficult")

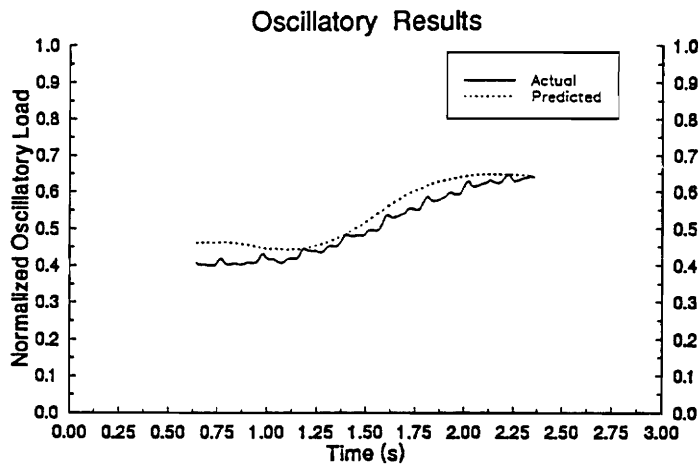
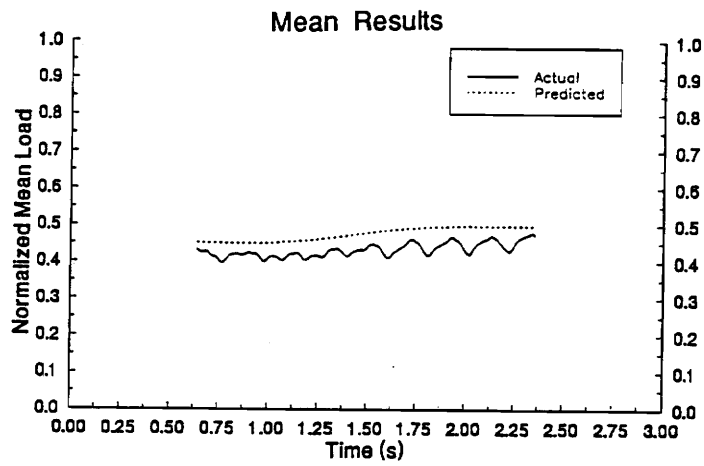
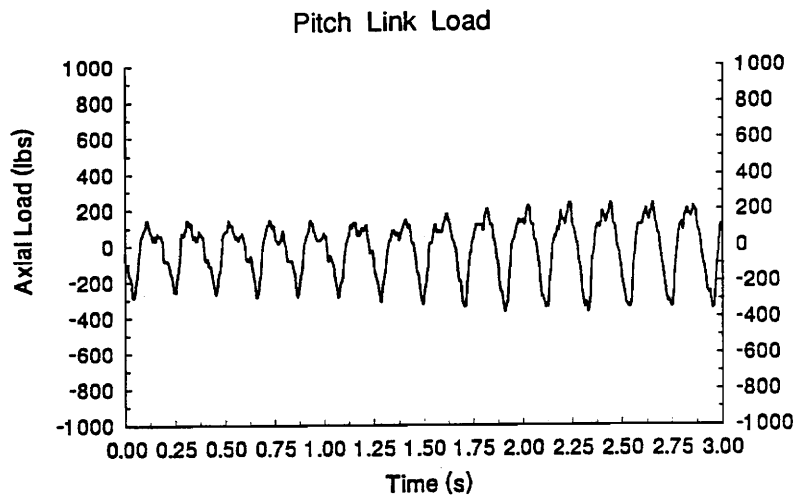


Figure 31. Pitch Link Results: Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, _____, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-two, "easy")

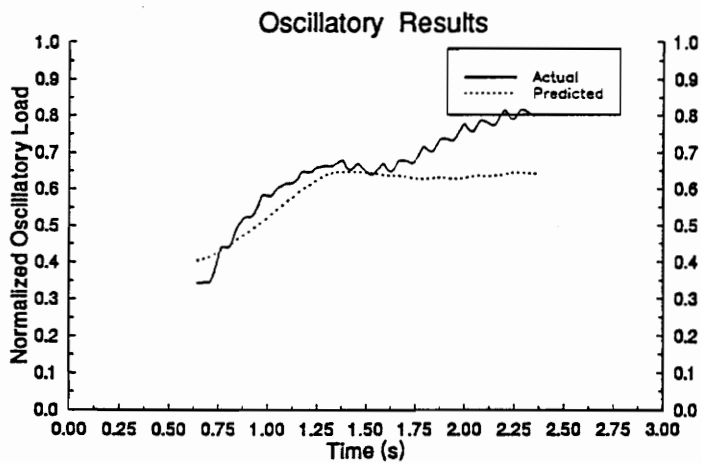
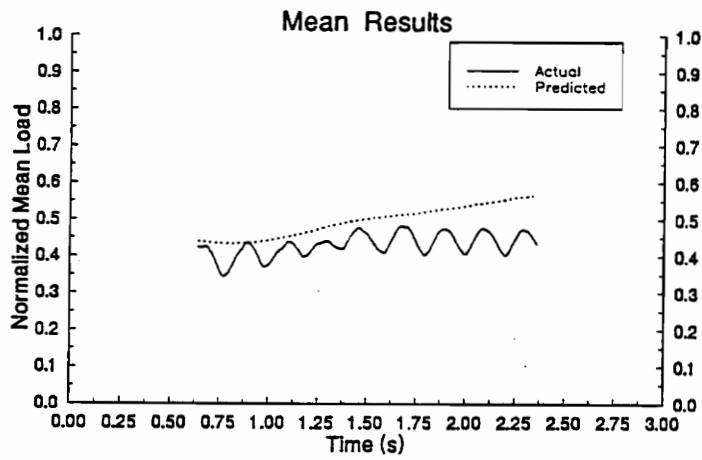
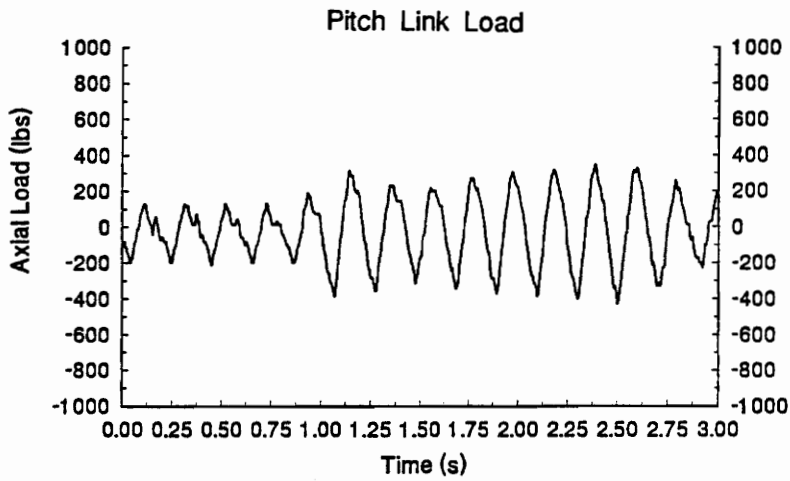


Figure 32. Pitch Link Results: Test Maneuver, +1.0Y, Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, ____, -1.0Y (mode-two, "difficult")

4.2.3 Lead-Lag Damper Results

Table 8 lists the training maneuvers and test maneuvers with the calculated average errors for the following lead-lag damper results. The average errors for the lead-lag damper predictions are significantly higher than those of the previous two components. This is addressed in the discussion of results. The format of Table 8 is the same as the previous two tables; the first two rows correspond to mode-one analysis, and the last two rows correspond to mode-two analysis; the first and third rows correspond to testing on the "easy" maneuver (-0.5X), and the second and fourth rows correspond to testing on the "difficult" maneuver (+1.0Y). Plots of the lead-lag damper results appear in Figures 33 thru 36, as listed in Table 8. Each figure contains the original load history and two sets of results.

Table 8. Lead-Lag Damper Test Results

Figure Number	Training Maneuvers	Test Maneuver	Avg Error (%) Mean Predictions	Avg Error (%) Oscillatory Predictions
33	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	2.8	2.6
34	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y	+1.0Y	5.3	5.2
35	+0.5X, +1.0X, _____, -1.0X, +0.5Y, +1.0Y, -1.0Y	-0.5X	9.9	13.3
36	+0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, _____, -1.0Y	+1.0Y	12.4	11.6

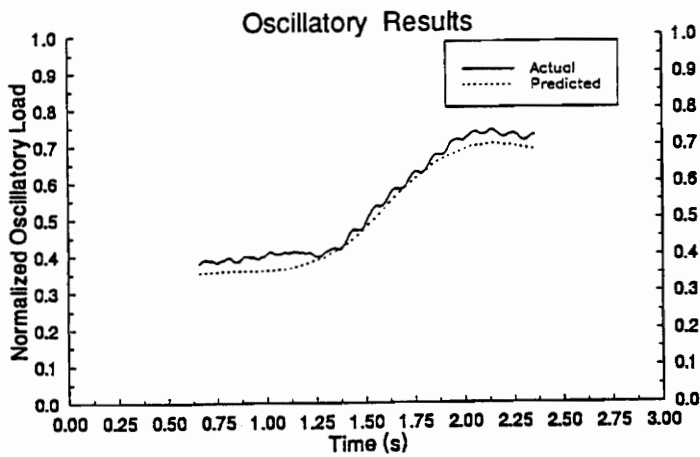
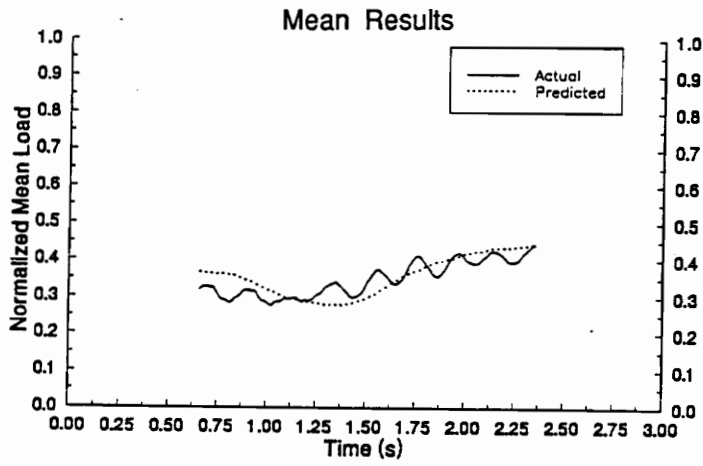
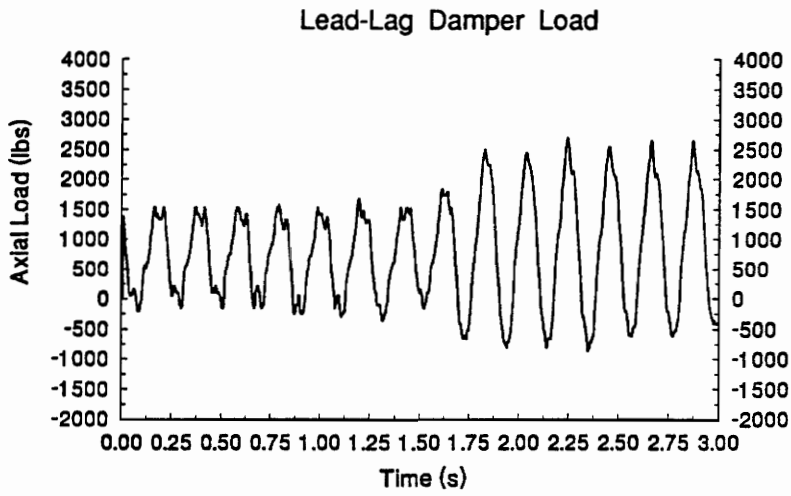


Figure 33. Lead-Lag Damper Results: Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "easy")

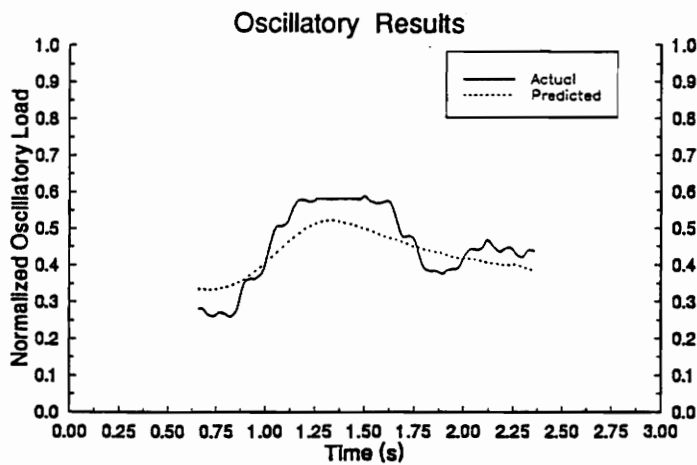
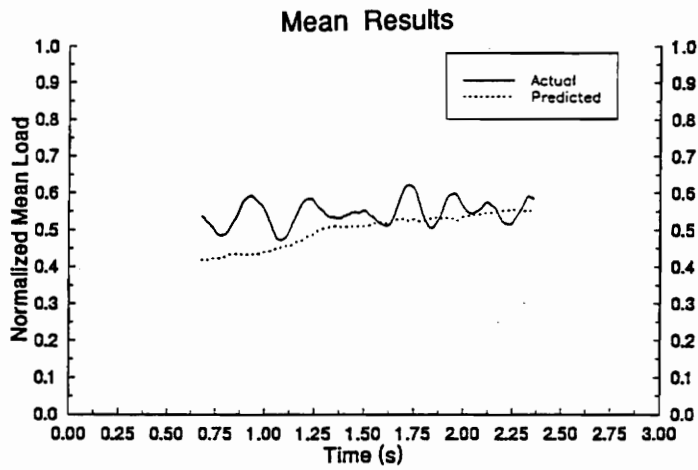
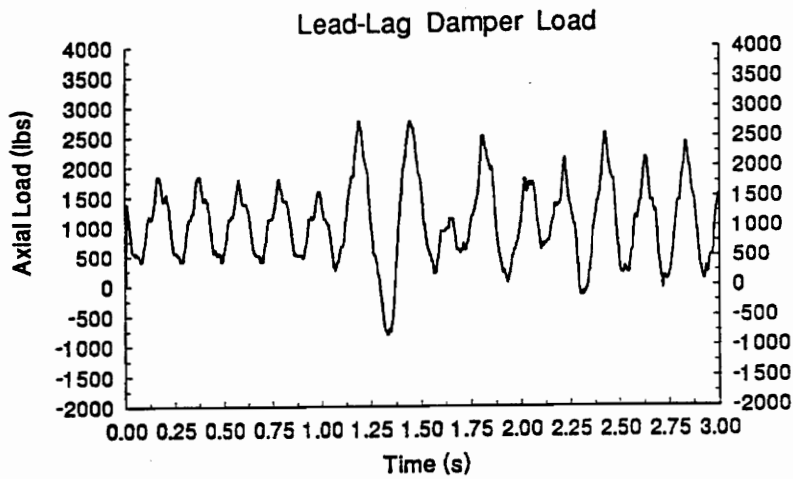


Figure 34. Lead-Lag Damper Results: Test Maneuver: +1.0Y. Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-one, "difficult")

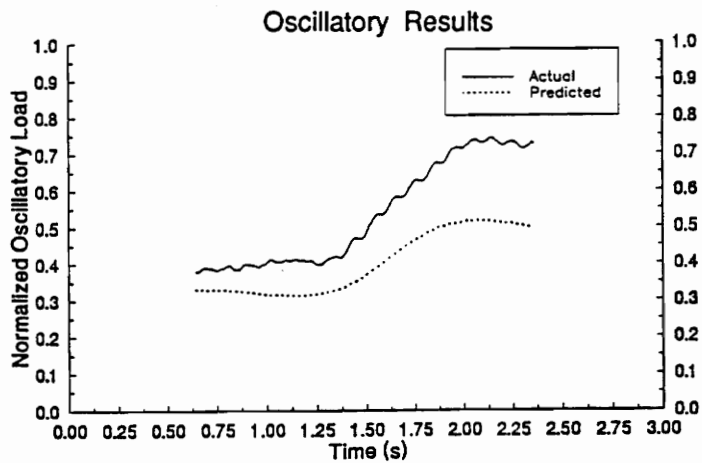
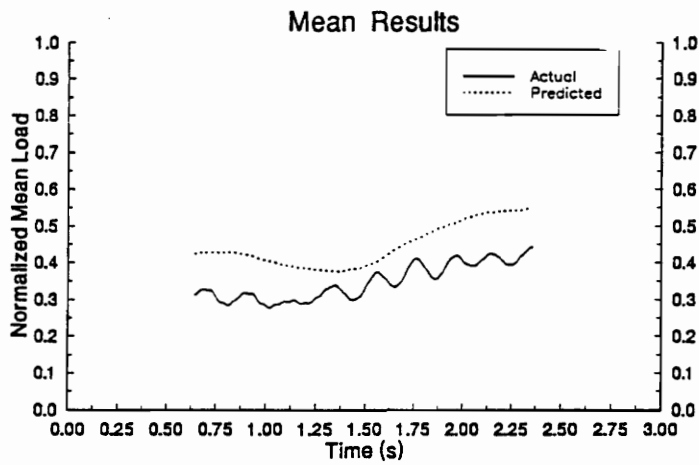
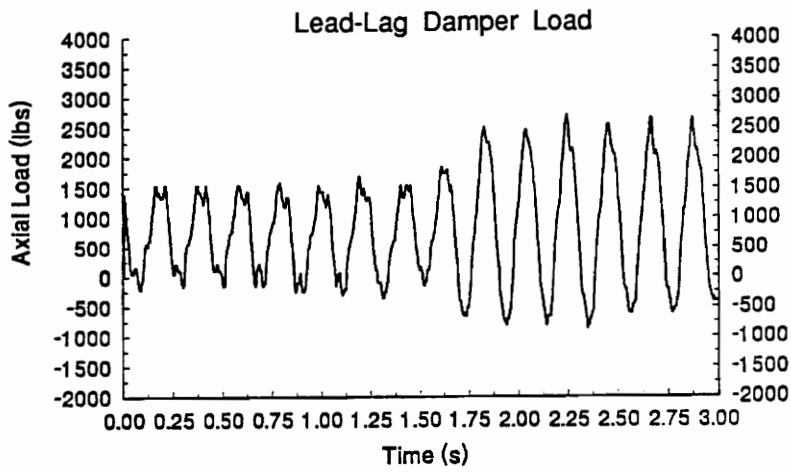


Figure 35. Lead-Lag Damper Results: Test Maneuver: -0.5X. Training Maneuvers: +0.5X, +1.0X, _____, -1.0X, +0.5Y, +1.0Y, -1.0Y (mode-two, "easy")

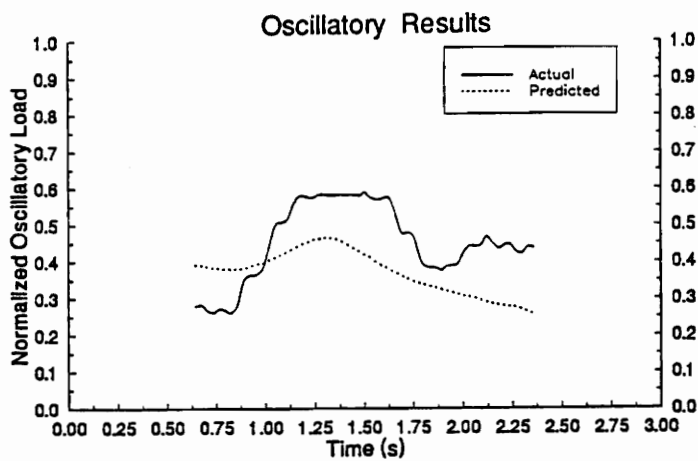
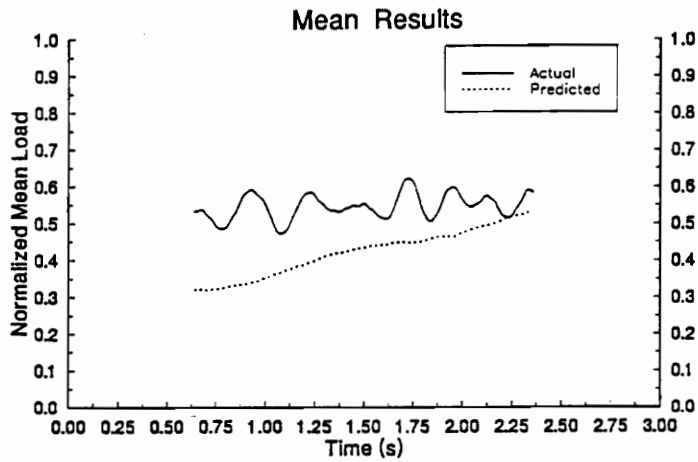
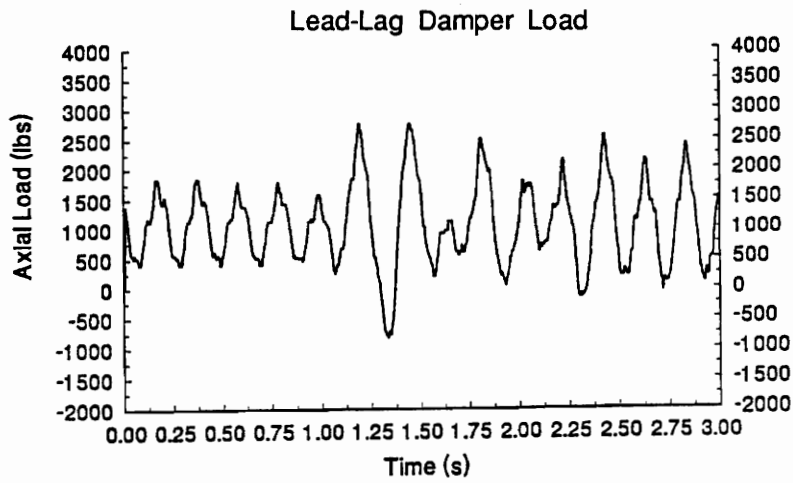


Figure 36. Lead-Lag Damper Results: Test Maneuver, +1.0Y, Training Maneuvers: +0.5X, +1.0X, -0.5X, -1.0X, +0.5Y, ____, -1.0Y (mode-two, "difficult")

4.3 Discussion of Results

Generally speaking, there is good qualitative and quantitative agreement between the predicted and measured loads for all cases. The range of average percent-of-scale errors is 1.1% to 13.3%. It is well worth mentioning again that, for each plot, the network was trained over several maneuvers. Therefore, in each case, the presented result is only one sample of the behavior of the network model.

The results of mode-one testing (Figures 25/26, 29/30, and 33/34) generally show excellent agreement between predicted and actual load constituents. These figures demonstrate the neural network's ability to accurately model loads when tested on data that is included in training.

Although not as accurate as mode-one predictions, the mode-two results (Figures 27/28, 31/32, and 35/36) generally show good qualitative agreement. Figures 27, 31, and 35 demonstrate the network's ability to interpolate between training maneuvers while Figures 28, 32, and 36 quantify the network's ability to extrapolate beyond the range of training maneuvers. In mode-two testing, the network generally predicts correct trends better than accurate magnitudes.

Another general observation is that the average errors are generally greater for the "difficult" maneuvers than for the "easy" maneuvers, as anticipated. This supports the presumption that some maneuvers are more difficult to model than others.

A more general observation from all of the plotted results is that the neural network appears to produce a filtered version of the desired load history. This observation is well demonstrated in Figure 33 where the modeled load constituent histories are smoother than the actual conditioned load constituent histories. The oscillations in the conditioned load histories are due to the small window size used in filtering the original signal. The network "filtering" phenomenon reflects both the characteristics of the input variables, and the underlying characteristics of the load-predicting neural network. The eight flight variables that were used as inputs are not periodic in nature. This, combined with the fact that the present network has no internal periodic wave generating capacity, gives the network no means to predict a periodic output. Finally, the network has been set up in such a way that it is trained and tested on

individual cases of data without regard for past or future cases; Each training case represents the magnitudes of each input variable at a specific instant in time, and the order that the cases are presented is random. Therefore, time-relevant information is lost. For this reason, the oscillatory nature of the conditioned output variables is treated as noise by the network.

4.3.1 Discussion of Tailboom Results

The range of average errors for the tailboom load predictions is 1.1% to 9.3%. The best case corresponds to the oscillatory predictions of Figure 25 (mode-one, "easy"). The worst case corresponds to the mean predictions of Figure 28 (mode-two, "difficult").

Mean prediction errors for the tailboom results are generally larger than the oscillatory prediction errors. This is because the range of mean loads was greater than that of oscillatory loads. The greater range of loads is more difficult to model, but the oscillatory prediction results are therefore more interesting.

A favorite plot among the tailboom results is the mean load history prediction of Figure 26 which corresponds to mode-one testing on the "difficult" maneuver. This plot demonstrates the network's ability to accurately model a fairly complicated load when it is included in training. This result shows the benefits of fully representing the range of system operation in the training data.

The mode-two tailboom results are not as accurate as mode-one. However, mode-two results more effectively prove the network's ability to understand underlying relationships rather than memorize training data. As expected, the performance of the network is degraded when the test maneuver is removed from the training data. This is most apparent when the mean predictions from both modes of testing on the "difficult" maneuver are compared (Figures 26 and 28). Here the average percent-of-scale error increased from 2.2% to 9.3%. As mentioned earlier, the network is required to extrapolate when testing in mode-two on +1.0Y. This, again, is because the test maneuver (+1.0Y) is beyond the range of the six training maneuvers. The implication, therefore, for mode-two testing on the "difficult" maneuver, is that the network prediction range has been extended beyond the training envelope. And, although this

is the worst case for tailboom predictions, the average error is still only 9.3%. Also, as shown in the plot, the network predicted the proper trends and approximate magnitudes.

Another notable characteristic of the mode-two results shown in Figures 27 and 28 is the tendency for the neural network predictions to be biased towards mid-range when testing on non-training data. This observation is most apparent in the mean load predictions. A possible explanation is the distribution of training cases. It was noticed, in general, that the distribution of the conditioned loads tended to centralize about 0.5. Therefore it is reasoned that, when faced with unfamiliar input information, the network is biased towards the most frequently represented training cases. One possible solution to this phenomenon would be the use of a large tolerance term in training so that redundant training cases can be ignored. Another method, having a similar effect, would be the removal of redundant or over-represented cases from the training data.

Perhaps a more surprising observation from the tailboom results is the extremely low, and similar, average errors for the results shown in Figures 25 and 26. Average errors for the "easy" maneuver are 2.3% and 1.1% (mean and oscillatory) while those of the "difficult" maneuver are 2.2% and 3.6%. It was earlier proposed that the +1.0Y maneuver loads would be more difficult to predict, but the average errors for these figures are nearly the same. The only explanation for this phenomenon is that, when tested on training data, the load-predicting network was able to accurately repeat the loads corresponding to the inputs. In other words, the network was not challenged beyond its' capacity. Also, it is somewhat irrelevant to compare the average test errors of Figures 25 and 26 because they are near to the tolerance threshold specified during training. The tolerance term allowed the network to ignore trivial cases such that errors below the tolerance caused no updates of internal weights.

However, when testing in mode-two, the difficulty anticipated for the +1.0Y maneuver is more evident. This is shown by the increase in average errors from Figure 27 to Figure 28. As shown in Table 6, mean and oscillatory prediction errors increased from 6.9% to 9.3% and 2.3% to 5.2% respectively when testing on the "difficult" maneuver as compared to testing on the "easy" maneuver.

Finally, upon examination of the mean tailboom prediction results from all four figures, it is noted that changes in the predicted loads seem to consistently precede the changes in the measured loads by a repeatable time shift. This is best seen in the mean prediction results of Figures 26 and 28. In particular, the predicted mean load history changes from a negative to positive slope around $t=1.30$ seconds while the actual mean load does not change in slope until about 1.50 seconds.

This observed time shift is attributed to the nature of the input variable records. As seen previously in Figures 23 and 24, the resultant variables such as pitch and roll rates lag behind the control variables such as stick position. This time difference was attributed to the response time of the aircraft. It is theorized that the neural network used both types of parameters to predict the tailboom load, which resulted in a prediction time shift that is a fraction of the helicopter response time. Thus, because the tailboom load is related to the stick position, but the network has no capacity to implement a time delay, the resulting loads from a stick input are predicted early.

4.3.2 Discussion of Pitch Link Results

The plotted pitch link results generally show good agreement between modeled loads and measured loads. It is again worth recalling that the network was trained with data from several maneuvers in each case. This helps to explain why the predicted loads for one maneuver do not exactly follow the measured trend, as in the center plot of Figure 29. The discrepancy in this plot is attributed to the network compromising its' solution with other maneuvers, similar to a least-squares curve-fit. Figure 36 shows testing results from the most difficult case.

The trends observed in the pitch link results are somewhat similar to those observed for the tailboom. The biggest exception is that the oscillatory constituents contained larger variations. This is because the pitch link loads are chiefly cyclic for these maneuvers. However, this would also lead one to expect the mean predictions to be highly accurate, which is not necessarily the case. For example, in Figure 29, the mean predicted load is consistently higher than the actual mean load. This is a very curious phenomena. One possible explanation is that the input variables contained no information that

could be related to this constituent of the load and the best solution the network could produce was a constant which best suits all of the training maneuvers.

Also peculiar is the fact that, for this same maneuver when tested in mode-two, the results were nearly identical. This partially confirms the previous theory that the network solution was a nearly constant output for all maneuvers.

The mode-two results for the "difficult" maneuver are more similar to the tailboom results. The mean and oscillatory average errors increased from 3.1% to 6.5% and 2.8% to 6.8% respectively (Figures 30 and 32). This, again, is to be expected when the test maneuver is removed from the training data. But still, the average errors of 6.5% and 6.8% are relatively small when considering that the network was predicting loads for a maneuver outside of the training envelope. This, again, demonstrates the network's ability to extrapolate information from the training maneuvers.

The oscillatory results of Figure 31, which correspond to mode-two testing on the -0.5X maneuver, demonstrate the network's ability to interpolate between training maneuvers -1.0X and +0.5X without a significant degradation in performance as compared to the similar test in mode-one Figure 29.

A final observation from the pitch link results is the lack of the time shift between the predicted and actual loads that was observed in the tailboom results. This makes sense because the pitch link loads should be directly related to the stick position variables with little or no time lag, as opposed to the tailboom load which is delayed by the response time of the aircraft. A step change in stick position causes a nearly simultaneous proportional change in the component loads in the rotating system. This observation is best demonstrated by the mode-one oscillatory results shown in Figures 29 and 30.

Discussion of Lead-Lag Damper Results

The plotted lead-lag damper results also demonstrate the capabilities of the load-predicting neural network. The relatively large errors of 12.4% and 11.6% for the "difficult" case predictions, as seen in Figures 34 and 36 are more easily understood on inspection of the original load. Here, it is

evident that the lead-lag damper loads are perhaps more complicated than the tailboom and pitch link loads. Additional and/or different inputs may be required for better predictions.

Figures 35 and 36 again show the effects of removing the test maneuver from the training data. Here the average percent-of-scale error for the oscillatory predictions increased from 2.6% to 13.3%. However, similar to results for the other two components, the predicted load follows the correct trend.

The hypothesis that mode-two predictions tend to err towards the center of the range is also supported by the lead-lag damper results. This is particularly noticeable in Figure 35 . In this test, the network is interpolating a solution which captures the trends fairly well but misses the magnitudes.

The least attractive results occurred for the mode-two "difficult" maneuver predictions, as shown in Figure 36. This may represent a case where the network was "over-trained". This would indicate that the network did not generalize a solution, but rather generated a solution that was limited by the range of training maneuvers. Extrapolation is not a normal virtue attributed to neural networks [6].

4.4 Secondary Observations

The following observations came about through the development and testing of the load-predicting neural network. For example, it was mentioned earlier that a method of feature selection was in development. The next section consists of preliminary results from that work where the relevance of the neural network input parameters was assessed.

4.4.1 Relevance of Input Parameters

Previously, in Section 3.3.1 the different methods of feature selection were introduced and briefly described. In relation to the loads prediction, one particular method was found to be most useful. To implement this method, a network is trained with an initial group of input variables and then tested. Next, individual input parameters are nullified by temporarily setting their connection weights to zero. The network is then tested without this parameter and the error compared to the original network error. The increase in error quantifies the relevance of the removed variable, as in equation (23):

$$relevance_m \propto \bar{E}_{without_m} - \bar{E}_{with_m}. \quad (23)$$

This process can be repeated for other input variables or combinations by restoring the nullified variable and removing the next variable(s). After assessing relevance of the all inputs to the trained network, one can re-train the network, replacing the least relevant parameters with new choices.

This method of quantifying relevance was used to assess the relevance of the eight flight variables chosen for predicting loads. Table 9 shows the relevance of each input parameter with respect to predicting the normalized time-varying mean bending load on the tailboom. These numbers reflect the relevance with regard to all seven maneuvers when assessed in mode-one. The significance of these results lies in the profile of the numbers with respect to one another, rather than the overall magnitude.

Table 9. Relevance of Input Parameters

Input Parameter	Relevance
Pitch Rate	13
Roll Rate	29
Yaw Rate	23
Vertical G's	59
Lateral G's	92
Longitudinal G's	33
Longitudinal Control Position	111
Lateral Control Position	30

These results show the longitudinal control position to be the most relevant parameter for predicting mean vertical tailboom bending. This supports the intuitive idea that the tailboom is like a

cantilever beam supporting the tail-rotor. Thus, the longitudinal control should directly effect bending in the tailboom.

A less intuitive result is the very low relevance of the pitch rate. Here, one might expect changes in aircraft pitch to be highly related to bending loads on the tailboom. In fact, as shown previously in Figure 24, the pitch rate does appear to be a significant input because of the large negative slope. However, a time delay exists between loading of the tailboom and changes in pitch. This low relevance and perceived delay could be dealt with by either discarding this parameter or introducing time dependency in the network. The relevance values for the other variables are even less intuitive, thus confirming the need for effective feature selection.

Another observation from this table of relevance is that three of the eight parameters are much more significant than the other five. This may imply that the five less-relevant parameters should be replaced, or perhaps even discarded and the network retrained to see if performance is impaired.

4.4.2 Variations in Network Geometry

In developing the load-predicting neural network, variations such as the number of outputs, the type of output unit, and the number of hidden layer units were investigated. Initially, the load-predicting network classified loads in a binary fashion using several nonlinear output units. The number of units activated above a given threshold indicated the load magnitude. This method was discarded after testing a single continuous output unit which proved more effective. Another variation was the change from a sigmoid to a linear output function. When using the standard sigmoid function, the network had difficulty predicting mid-range values. The linear unit enhanced predictions throughout the range of zero to one. Finally, a network having two linear output units was tested. The goal was to train a single network to predict both mean or oscillatory components of a load. However, reduced overall performance compared to separate networks for each component.

The number of units in the hidden layer of a neural network usually indicates the level of complexity in a system that it can effectively model. However, more units are not necessarily better;

excessive units may prevent convergence and also slow performance. In developing the load-predicting neural network, it was found that four units in the hidden layer were sufficient. In some cases, the network performance with only two hidden layer units was within 2 or 3% of performance with four units. Using more than four units significantly increased the training time while only marginally improving performance, if at all. These findings led to the general rule for this data set that four hidden layer units is adequate.

Finally, the number of layers between the input and output layers can be varied. Most references agree that a three-layer network is sufficient for modeling complicated nonlinear relationships. However, four-layer networks (those having two hidden layers) are also popular and are utilized when three-layer nets fail to perform satisfactorily. For predicting helicopter loads, the three-layer geometry was chosen because it trains significantly faster and it was felt that it had not been tested beyond a satisfactory level of performance.

4.4.3 Variations in Training Specifications

Variations in the back-propagation training algorithm were also examined. These included adding tolerance and experimenting with a momentum term. Tolerance in training is intended to allow a specified amount of error for the more redundant cases in hopes of achieving improved performance for the limiting cases. With respect to loads prediction, it was found that tolerances on the order of 5% did not significantly improve performance for the extreme cases. However, the use of tolerance greatly reduced training time by allowing the algorithm to skip negligible weight updates corresponding to trivial cases. As the network learned to model the system, the percent of training cases that caused an update in the connecting weights decreased rapidly. In one case, a tolerance of 4% resulted in only about 10% of the training cases effecting a weight update when training was nearly complete.

The use of a momentum term for training is intended to speed convergence by adding a fraction of the past weight change to the current weight change for each connection. This option was tested with

the load-predicting network but failed to improve performance. Momentum was therefore left out of the training algorithm.

Finally, variations in the learning rate within the established range of 0.3 to 0.6 had little effect on the final network performance. It was evident that any value within this range would eventually result in approximately the same level of performance if given enough training exposure. Use of the upper limit resulted in faster convergence while the lower rate required more iterations. It was found that learning rates greater than 0.7 occasionally resulted in unstable (and undesirable) error behavior.

5 Conclusions

Continued pursuit of the neural network method of indirectly monitoring dynamic structural helicopter loads from flight variables has been justified. A three-layer load-predicting neural network has been designed to resolve time-varying mean and oscillatory load histories from eight commonly monitored flight variables. The implementation of this design included modification of the popular back-propagation training algorithm and development of effective data preprocessing and manipulating programs.

The network successfully modeled the nonlinear relationships between the eight flight variables and the loads on three critical components over an envelope of seven standard maneuvers. The first load considered, bending in the vertical plane of the tailboom, occurs in the fixed (airframe) system. The other two loads modeled, axial loading of a pitch link and a lead-lag damper, occur in the rotating main-rotor blades system. Predictions of both time-varying mean and oscillatory load constituents ranged from approximately 90 to 100 percent-of-scale accurate.

The predictive ability of the network was analyzed over this set of seven standard maneuvers, which consisted of step changes in the control-stick position. Results from two representative maneuvers were shown. The selected maneuvers were labeled "easy" and "difficult" in accordance with their corresponding location in the stick position coordinate space.

Testing was executed in two modes. In mode-one tests, the network was tested on data that was included in training. The most accurate predictions corresponded to testing in mode one over the "easy" maneuver, as expected. The high level of agreement for these results demonstrated the neural network's ability to accurately mimic behavior contained in the training data.

In mode-two, the network was tested on non-training data. This mode was more challenging. The prediction results better qualified the network's ability to generalize and demonstrate an understanding of the underlying nonlinear input/output relationships. Although agreement between predicted and measured load constituents was lower for mode-two tests, the modeled loads showed generally correct trends. The least accurate predictions corresponded to mode-two testing over the "difficult" maneuver, as also expected.

When testing in mode-two over the "difficult" maneuver, the network was operating outside of the range of training data and was therefore required to extrapolate from the encoded knowledge. Although these results were numerically the poorest among those presented, they warranted greater attention. In considering the helicopter loads application, it is unrealistic to expect a training data-base to contain the entire spectrum of possible maneuvers. Therefore a model which demonstrates both interpolative and extrapolative qualities is most favorable. Even if the extrapolative capabilities of the model are limited, they still exceed the current methods of load monitoring.

The selection of input variables was based on engineering judgement and availability. The rudimentary work of implementing an automated "feature selector", which determines the relevance of individual input parameters, has been completed and demonstrated.

A range of effective training specifications was established for the number of hidden layer units, the learning rate, the number of iterations, and the tolerance. Typically the network consisted of four hidden layer units and was trained with a learning rate of 0.4 and a tolerance of 2% for 500 passes through the training data. Variations within the established range of training specifications had no significant end effect on the performance of the trained network.

6 Recommendations

In order to expedite further investigation of the neural network application to loads prediction for helicopters, the following recommendations are given:

The range of maneuvers which the current load-predicting network has been tested on should be greatly expanded. This will require the development of more sophisticated file manipulating and managing techniques. This level of agreement demonstrated in the results validates the persual of more difficult cases.

It is recommended that the neural network be modified to predict the entire load history, as opposed to the time-varying mean and oscillatory statistical quantities. Because the loads are not purely sinuosoidal, these quantities cannot be recombined to effectively create an precise load history. Accurate knowledge of the load history is necessary for fatigue damage assessment. Prediction of the entire load history will require the development of more sophisticated and innovative neural network structures and accompanying training algorithms. To accomplish prediction of the entire signal, the following suggestions are given:

- 1) The inclusion of an internal periodic function generator in the neural network with adaptive amplitude and phase should be investigated.
- 2) The incorporation of linear feed-thru connections from the input layer directly to the output should be investigated.

- 3) The investigation of various squashing function types within the network structure should be investigated.
- 4) The generating of additional inputs as functions of the original inputs and/or combinations of the original inputs (i.e. products, squares, roots, etc) should be investigated.
- 5) The incorporation of time dependency in the neural network through time delays, and the calculation of time varying first and second derivatives for additional inputs should be investigated.
- 6) Sophisticated methods of studying the trained neural network including graphics capabilities should be developed. These include the ability to quantify the nonlinearity of the model and the level of activity associated with each interconnection.

Other suggestions for improving the efficiency of an in-depth investigation are the continued development of an automated feature selector and the development of efficient methods for managing large data files.

References

1. Elliott, S. J., and P. A. Nelson. "The Active Control of Sound," *Electronics & Communication Engineering Journal*. August 1990. pp. 127-136.
2. Johnson, R. B., G. L. Martin, and M. S. Moran. "A Feasibility Study for Monitoring Systems of Fatigue Damage to Helicopter Components," USAAMRDL-TR-74-92. January 1975.
3. Obermeier, K. K., and J. J. Barron. "Time to Get Fired Up", *B Y T E*. August 1989. pp. 217-224.
4. Lippmann, R. P. "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*. April 1987. pp. 4-21.
5. Bailey, D. L., and D. M. Thompson. "Developing Neural Network Applications," *AI Expert*. September 1990. pp. 34-40.
6. Vanlandingham, H. "Adaptive and Intelligent Controls," Graduate Course in Electrical Engineering. VPI&SU. Fall 1991.
7. Nelson, M. M., and W. T. Illingworth. *A Practical Guide to Neural Nets*. Addison-Wesley Publishing Company, Inc. 1991.
8. Korkosz, G. J. "Methods of Full Scale Fatigue Testing the Apache Helicopter Rotor Systems," Proceedings of the AJS Midwest Region Helicopter Fatigue Specialists' Meeting. St. Louis, Missouri. October 1984.

9. Harrington, J., and R. D. Roesch. "Air-to-Air Combat Test IV (AACT IV) Volume II - AH-64 Structural Analysis," USAAVSCOM TR-88-D-18B. November 1989.
10. Buckner, R. private communications. Ft. Eustis, Virginia. July-December 1990.
11. Gunsallus, C. T., C. P. Hardersen, and P. G. Stennett. "Investigation of Fatigue Methodology," USAAVSCOM TR-87-D-17. May 1988.
12. Ryan, J. P., A. P. Berens, R. G. Coy, and G. J. Roth. "Helicopter Fatigue Load and Life Determination Methods," USAAMRDL TR-75-27. August 1975.
13. Cabell, R. H. private communications. July-December 1990.
14. Arden, R. W. "Hypothetical Fatigue Life Problem," Proceedings of the Specialists Meeting on Helicopter Fatigue Methodology Sponsored by the Midwest Region of the American Helicopter Society. St. Louis, Missouri. March 1980.
15. Mozer, M. C., and P. Smolensky. "Skeletonization: A Technique for Trimming the Fat from a Network Via Relevance Assessment," (Technical Report CU-CS-421-89). Boulder: University of Colorado, Department of Computer Science. 1989.
16. Cabell, R. H. "The Automatic Identification of Aerospace Acoustic Sources," M.S. Thesis. VPI&SU, Blacksburg, VA. February 1989.
17. Scott, E. A. "Recognition of Aerospace Acoustic Sources Using Advanced Pattern Recognition Techniques," M.S. Thesis. VPI&SU, Blacksburg, VA. February 1991.

Vita

Allan Binford "Ford" Cook was born at the United States Air Force Academy, Colorado Springs, Colorado on May 13, 1966. Ford graduated from high school in Nashville, Tennessee in May 1984. After one year of engineering at the University of Tennessee, Ford transferred to Va Tech and received his Bachelor of Science in Mechanical Engineering in the spring of 1989. Ford worked as an engineer in the Combustion Laboratory at Va Tech for eight months after graduation before returning to school to pursue a Masters degree in Mechanical Engineering. Upon completion, Ford plans to gain practical experience as a professional engineer before furthering his formal education.

A handwritten signature in cursive script that reads "Allan Binford Cook". The signature is written in black ink and is positioned above a horizontal line.

Allan Binford Cook